НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

«На правах рукопису»

«До захисту допущено»
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
“___”_____2019 р.

УДК 004.912

Магістерська дисертація

зі спеціальності *121 «Інженерія програмного забезпечення»*

на тему: «*Математичне та програмне забезпечення реферування тексту*»

Виконав: студент VI курсу, групи *ІП-з81мп*
Аршакян Георгій Давидович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник *ст. викл. Олійник Ю.О.* _____
(посада, науковий ступінь, , вчене звання, прізвище та ініціали) (підпис)

Консультант *доц., к.т.н., Ліщук К.І.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент *доц. каф. ТК, к.т.н., доц. Ткач М.М.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань

Національний технічний університет України

«Київський політехнічний інститут

імені Ігоря Сікорського»

Факультет *інформатики та обчислювальної техніки*

(повна назва)

Кафедра *автоматизованих систем обробки інформації та управління*

(повна назва)

Рівень вищої освіти *другий (магістерський) за освітньо-професійною програмою*

Спеціальність *121 «Інженерія програмного забезпечення»*

(код і назва)

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ *О.А. Павлов*

(підпис) (ініціали, прізвище)

«___» _____ 201 р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Аршакяну Георгію Давидовичу

(прізвище, ім'я, по батькові)

1. Тема дисертації *Математичне та програмне забезпечення реферування тексту*

науковий керівник дисертації *ст. викл. Олійник Ю.О.*

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “ 5 ” *листопада* 2019 р. № 3836-с

2. Строк подання студентом дисертації “ ” 20 р.

3. Об’єкти дослідження *математичні моделі автоматичного ерування тексту*

4. Предмет дослідження *є розробка застосунків, що будуть використовувати ючі методи або нові методи автоматичного реферування текстів, исаних українською та російською мовами*

5. Перелік завдань, які потрібно розробити *вивчення математичних та програмних методів реферування тексту; постановка формальної задачі; удосконалення програмних методів реферування текстів*

6. Перелік графічного матеріалу

Схема роботи алгоритму,

Діаграма класів розробленого застосунку

Діаграма розгортання розробленого застосунку

Діаграма послідовностей

7. Орієнтовний перелік публікацій *Огляд підходів та методів автоматичного еферування тексту; Text stream sentiment analysis with summarization reprocessing;*

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання	завдання

		видав	прийняв

9. Дата видачі завдання “ *01* ” *вересня* 20 *19* р

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	*Формалізація результатів огляду існуючих математичних методів реферування текстів*	*02.09.2019*	
2	*Порівняльний аналіз існуючих методів реферування та аналіз існуючих програмних рішень*	*01.10.2019*	
3	*Постановка та формалізація математичної моделі задачі*	*16.10.2019*	
4	*Імплементація та модифікація існуючих методів екстрактивних методів реферування*	*31.10.2019*	
5	*Розробка програмного забезпечення*	*15.11.2019*	
7	*Проведення експериментальних досліджень розробленого алгоритму*	*30.11.2019*	
8	*Оформлення документації*	*04.12.2019*	
9	*Подання роботи на попередній захист*	*05.12.2019*	
10	*Подання роботи на основний захист*	*16.12.2019*	

Студент

(підпис) (ініціали, прізвище)

Науковий керівник

(підпис) (ініціали, прізвище)

РЕФЕРАТ

Магістерська дисертація: 96 с., 10 рис., 11 табл., 5 додатків, 38 джерел.

Актуальність роботи заключається в тому, що кількість застосунків, що вирішують задачу автоматичного реферування текстів, написаних українською чи російською невелика, та результати їх роботи не можна назвати дуже ефективними з точки зору математичних показників лексичної близькості текстів. Але задача реферування текстів є досить важливою зважаючи на кількість потоків текстових даних, які стають більшими кожен день і для багатьох професій – від законотворців, журналістів та, навіть, до військових критично ці потоки ефективно обробляти використовуючи менше часу. Аналогічно, реферування є важливим і для машинної обробки текстів – це значно пришвидшує обмін даними та має дуже багато потенційних аплікацій.

Мета дослідження є синтез нового математичного та програмного рішення для автоматичного реферування текстів, написаних українською та російською мовами.

Для реалізації поставленої мети були сформульовані **наступні завдання**:

- Аналіз існуючого теоретичного апарату реферування текстів
- Огляд існуючого програмного забезпечення
- Розробка нового застосунку та моделі для реферування текстів українською та російською мовами
- Виконати аналіз отриманих результатів

Об’єкт дослідження: є математичні моделі автоматичного реферування тексту.

Предметом дослідження є розробка застосунків, що будуть використовувати існуючі методи або нові методи автоматичного реферування текстів, написаних українською та російською мовами.

Методи досліджень є методи екстрактивного реферування текстів, що базуються на векторному представленні тексту

Наукова новизна: Найбільш суттєвими науковими результатами магістерської дисертації є розробка власного алгоритму реферування текстів та його імплементація для вирішення задачі реферування текстів, написаних українською та російською мовами.

Практичне значення отриманих результатів визначається тим, що запропонований алгоритм та розроблений програмний застосунок у результаті експерименту показує кращі результати за існуючі підходи.

Зв'язок роботи з науковими програмами, планами, темами: Робота виконувалась на кафедрі автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського» в рамках теми «Методи та технології високопродуктивних обчислень та обробки надвеликих масивів даних». Державний реєстраційний номер 0117U000924.

Публікації: Наукові положення дисертації опубліковані в тезах наукової конференції студентів, магістрантів та аспірантів «Інформатика та обчислювальна техніка» – ІОТ-2019[27]

АВТОМАТИЧНЕ РЕФЕРУВАННЯ ТЕКСТІВ, ЕКСТРАКТИВНІ МЕТОДИ РЕФЕРУВАННЯ, TEXTRANK, ЛАТЕНТНО-СЕМАНТИЧНИЙ АНАЛІЗ, МАТЕМАТИЧНІ МОДЕЛІ ПРЕДСТАВЛЕННЯ ТЕКСТІВ

ABSTRACT

Master dissertation: 96 pp., 10 figures, 11 tables, 5 applications, 38 sources. **Topicality** is concluded with that the number of applications that solve the problem of automatic abstract generation for the texts written in Ukrainian or Russian is extremely small, and the results of their work can not be called very effective in terms of mathematical indicators of the lexical proximity of texts. But the task of text summarization is quite important, especially given the number of textual data streams that are growing every day and for many professions - from lawmakers, journalists, and even the military – it's critical to process these streams faster. Similarly, abstracting is important for word processing - it greatly speeds up data exchange and has many potential applications.

The aim of the research is synthesis of a new mathematical and software solution for automatic abstracting of texts written in Ukrainian and Russian. To achieve this goal which were formulated **following tasks**:

- Existing theoretical methods of text summarization analysis
- Review of existing text summarization software
- Development of a new application and mathematical model for Russian and Ukrainian texts summarization
- Perform the analysis of the obtained results

The object of research is the mathematical models for automatic text summarization

The subject of research is the development of applications that will use existing methods or new methods of automatic text summarization of texts written in Ukrainian and Russian.

Research methods are the extraction methods of the text summarization based on the vector representation of the text

Scientific novelty of the obtained results. The most significant scientific results of the master's thesis are the development of its own algorithm for abstracting texts and its implementation to solve the problem of abstracting texts written in Ukrainian and Russian.

Practical consequences of the results are determined by the fact that the proposed algorithm and the developed software application as a result of the experiment show better results than the existing approaches.

Relationship of work with scientific programs, plans, themes: The work was performed at the Department of Automated Information Processing and Management Systems of the National Technical University of Ukraine «Kyiv Polytechnic Institute. Igor Sikorsky ”within the topic“ Methods and technologies of high-performance computing and processing of large data sets ”. State Registration Number 0117U000924.

Publications: The research results were published in the thesis of the scientific conference of students, undergraduates and graduate students "Computer Science and Computer Engineering" - IOT-2019[27].

AUTOMATIC TEXT SUMMARIZATION, EXTRACTIVE TEXT SUMMARIZATION METHODS, TEXTRANK, LATENT-SEMANTIC ANALYSIS, MATHEMATICAL MODELS OF TEXT REPRESENTATION

ЗМІСТ

ВСТУП

Сьогодення наповнене письмовою інформацією – згідно з підрахунками однієї відомої газети, для того, щоб роздрукувати усі веб-сторінки необхідно 305.5 мільярдів сторінок А4[1]. Кожного дня потоки даних текстів оточують людей, і кожного дня об’єм цих текстів збільшується. В таких умовах доречно розглянути таку математичну та програмну проблему як автоматична реферація текстів.

Системи реферування текстів сприяють зменшенню об’єму цих текстів із збереженням його основного змісту. Це однозначно зменшує кількість ресурсів, яку треба витратити людині чи машині для обробки цих текстів, процес пошуку та виявлення корисної інформації. Також важливо зазначити, що за допомогою методів реферування можна значно зменшувати розміри текстових наборів даних, які будуть використовуватись для тренування систем з машинним навчанням і тим самим потенційно економити ресурси, необхідні для тренування без збільшення помилок.

Завданням даної роботи є дослідження методів автоматичного реферування текстів через аналіз існуючого теоретичного апарату обробки природної мови, аналіз існуючих застосунків які реалізують методи реферування та розробка нового застосунку та моделі для реферування текстів українською та російською мовами

Метою даної роботи є синтез нового математичного та програмного рішення для автоматичного реферування текстів, написаних українською та російською мовами

Актуальність роботи заключається в тому, що кількість застосунків, що вирішують задачу автоматичного реферування текстів, написаних українською чи російською невелика, та результати їх роботи не можна назвати дуже ефективними з точки зору математичних показників лексичної близькості

текстів. Але задача реферування текстів є досить важливою зважаючи на кількість потоків текстових даних, які стають більшими кожен день і для багатьох професій – від законотворців, журналістів та, навіть, до військових критично ці потоки ефективно обробляти використовуючи менше часу. Аналогічно, реферування є важливим і для машинної обробки текстів – це значно пришвидшує обмін даними та має дуже багато потенційних аплікацій.

Завданням дослідження є вивчення математичних та програмних методів реферування тексту, зокрема латентно-семантичний аналіз, метод штучної нейронної мережі а також, удосконалення програмних методів реферування текстів.

Об'єктом дослідження є математичні моделі автоматичного реферування тексту.

Предметом дослідження є розробка застосунків, що будуть використовувати існуючі методи або нові методи автоматичного реферування текстів, написаних українською та російською мовами.

Методи дослідження. В основу досліджень покладено дві категорії методів реферування текстів – екстракція та генерування. Дослідження виконуються з використанням вже існуючого математичного апарату та методів обробки інформації.

Наукова новизна: Найбільш суттєвими науковими результатами магістерської дисертації є розробка власного алгоритму реферування текстів та його імплементація для вирішення задачі реферування текстів, написаних українською та російською мовами.

Практичне значення отриманих результатів визначається тим, що запропонований алгоритм та розроблений програмний застосунок у результаті експерименту показує кращі результати за існуючі підходи.

1 АНАЛІЗ ДЖЕРЕЛ

1.1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО АВТОМАТИЧНУ ОБРОБКУ ТЕКСТІВ

Поява мережі Інтернет та бурхливе зростання доступної текстової інформації значно прискорило розвиток наукової галузі, яка існує вже багато десятків років і відома як автоматична обробка текстів NLP (Natural Language Processing) і комп'ютерна лінгвістика (Computational Linguistics)[5]. В межах цієї області запропоновано багато перспективних ідей по автоматичній обробці текстів на природній мові (ПМ), які були втілені в багатьох прикладних системах, в тому числі комерційних. Сфера додатків комп'ютерної лінгвістики постійно розширюється, з'являються все нові завдання, які успішно вирішуються, в тому числі із залученням результатів суміжних наукових областей.

Про наукові досягнення області можна отримати уявлення по інтернет-сайту ACL (Association of Computational Linguistics)[2] - міжнародної Асоціації з Комп'ютерної Лінгвістики, на якому агрегуються роботи численних наукових конференцій в цій області.

Комп'ютерна лінгвістика (КЛ) - міждисциплінарна область, яка виникла на стику таких наук, як лінгвістика, математика, інформатика (Computer Science), штучний інтелект (Artificial Intelligence). У своєму розвитку вона до сих пір вбирає і застосовує (при необхідності адаптуючи) розроблені в цих науках методи і інструменти.

Витоки КЛ сходять до досліджень відомого американського лінгвіста Н. Хомського по формалізації структури природної мови[3], до перших експериментів з машинного перекладу, виконаних програмістами і математиками, а також до розроблених в області штучного інтелекту першим програмами розуміння природної мови (наприклад, [4]).

Оскільки в КЛ об'єктом обробки є тексти природної мови, її розвиток неможливий без базових знань в області загальної лінгвістики (мовознавства) [5]. Лінгвістика вивчає загальні закони природної мови - його структуру і функціонування, і включає такі області:

- фонологія - вивчає звуки мови і правила їх з'єднання;
- морфологія - займається внутрішньою структурою і зовнішньою формою слів мови, включаючи частини мови і їх категорії;
- синтаксис - вивчає структуру пропозицій, правила сполучуваності та порядку розташування слів у реченні, а також загальні його властивості як одиниці мови;
- семантика і прагматика - тісно пов'язані області: семантика займається змістом слів, пропозицій та інших одиниць мови, а прагматика - особливостями вираження цього сенсу в зв'язку з конкретними цілями спілкування;
- лексикографія описує лексикон конкретної ПМ - її окремі слова, їх граматичні і семантичні властивості, а також методи створення словників.

Найтісніше комп'ютерна лінгвістика пов'язана з областю штучного інтелекту (ШІ)[6], в рамках якого розробляються програмні моделі окремих інтелектуальних функцій. Незважаючи на очевидне перетинання досліджень в області комп'ютерної лінгвістики та ШІ (оскільки володіння мовою відноситься до інтелектуальних функцій), ШІ не поглинає всю КЛ, оскільки вона має свій теоретичний базис і методологію. Загальним для зазначених наук є комп'ютерне моделювання як основний спосіб і підсумкова мета досліджень, евристичний характер багатьох застосовуваних методів.

Дещо спрощено завдання комп'ютерної лінгвістики може бути сформульовано як розробка методів і засобів побудови лінгвістичних процесорів для різних прикладних задач по автоматичній обробці текстів на ПМ. Розробка лінгвістичного процесора для деякої прикладної задачі

передбачає формальний опис лінгвістичних властивостей оброблюваного тексту (хоча б найпростіших), який може розглядатися як модель тексту (або модель мови).

Реферування тексту (Summarization) - скорочення його обсягу і отримання короткого викладу його змісту - реферату, що пришвидшує пошук в колекціях документів. Реферат може складатися також для кількох близьких по темі документів (наприклад, по кластеру новин-документів). Основним методом автоматичного реферування досі є відбір найбільш значущих пропозицій реферованого тексту на основі статистики слів і словосполучень, а також структурних і лінгвістичних особливостей текстів[3].

Близьке до реферування завдання – анотування (Abstract Generation) тексту документа, тобто складання його анотації. У простій формі анотація являє собою перелік основних (ключових) тем тексту, для виділення яких використовуються статистичні та лінгвістичні критерії. При обробці великої колекції документів актуальні завдання класифікації (Categorization) і кластеризації текстів (Text Clustering)[7]. Класифікація означає віднесення кожного документа до певного класу із заздалегідь відомими параметрами, а кластеризація - розбиття безлічі документів на кластери, тобто підмножини тематично близьких документів. Для вирішення цих завдань застосовуються методи машинного навчання, в зв'язку з чим ці прикладні завдання часто відносять до напрямку Text Mining, оскільки він розглядався як частина наукової області Data Mining (інтелектуальний аналіз даних)[8].

Як бачимо, автоматична обробка текстів має дуже багато зв’язків з іншими науками[9], і формально для цієї роботи ми будемо використовувати лише дуже невеликий апарат для роботи з досить вузькою задачею реферування та анотування і далі будемо розглядати основні підходи для реферування текстів.

1.2 АНАЛІЗ ВІДОМИХ ПІДХОДІВ РЕФЕРУВАННЯ ТЕКСТІВ

Методи реферації текстів зазвичай можна поділити на два підтипи: методи, що не передбачають опору на знання, та методи з опорою на знання[17].

Методи без опори на знання зазвичай використовують різні статистичні показники (як, наприклад, TF-IDF чи TLTF), ігноруючи при цьому певні лінгвістичні та семантичні особливості природніх текстів, тому таке реферування є квазіреферуванням, що зводиться до екстракції з тексту елементів, що мають найменші показники релевантності. Ці методи зазвичай використовують певні вагові коефіцієнти для блоків тексту, які розраховуються зважаючи на частоту використання цього блоку, розташування у тексті, словникові особливості тощо[29].

Методи з опорою на знання в той час потребують набагато більше обчислювальних можливостей через використання інструментів автоматичної обробки текстів, словників, лексичних та синтаксичних аналізаторів. Окрім цього, для коректної роботи цих методів надважливо мати бази знань, в яких відображені поняття предметної галузі, які необхідні для визначення найбільш важливих речень. Основні методи з опорою на знання використовують традиційний синтаксичний розбір речень, при цьому також застосовується семантична інформація для анотування дерев розбору цих речень. Процедури порівняння маніпулюють безпосередньо деревами з метою видалення чи перегрупування їх частин, наприклад, шляхом скорочення гілок на підставі деяких структурних критеріїв, таких як дужки або вбудовані умовні чи підлеглі речення. Після такої процедури дерево розбору істотно спрощується, стаючи, по суті, структурною витримкою вихідного тексту[29].

Окрім розділення підходів до задачі реферування по критерію опори на знання, всі відомі методи можна розділити зважаючи на наступні критерії:

- методи екстракції та абстракції;

- методи реферації документу та реферації множин документів;
- методи загальної реферації текстів та реферації по запитам.

Екстрактивні та абстрактні методи зазвичай дуже відрізняються, так як використовують різні підходи до математичної задачі реферування[18]. Так, абстрактні методи зазвичай використовують методи машинного навчання та методи генерування текстів – так як, по суті вони створюють нову множину речень з початкової, чи новий документ з початкової множини – не використовуючи напряму існуючі документи – а проаналізувавши їх, згенерувати їх стислий зміст у вигляді нових сутностей. Задача автоматичного реферування з екстрактивними методами використовує інший підхід – за допомогою певних перетворень, через розрахунки статистичних показників, вагових коефіцієнтів з документу чи множини документів ми залишаємо лише значущі елементи в тексті (можливо, застосовуючи певні лексичні та синтаксичні перетворення для кращої якості).

Методи екстракції на сьогоднішній день майже повністю досліджені та імплементовані, в той час, як абстрактні методи дуже стрімко розвиваються, так як необхідний математичний апарат та обчислювальні можливості дозволяють для досить вузьких задач давати для цих методів кращі результати[18].

Методи ж реферації множин документів відрізняються від методів для реферації одного документу можливістю обрати два різних за собою підходи до роботи алгоритму – конкатенація множини документів у один та вирішення задачі реферації відомими методами, чи виділення рефератів у кожного окремого документа та використання методів для виділення з рефератів загального. Саме ця група методів зараз стрімко розвивається так як потенційно може давати кращі результати аніж проста конкатенація[18].

Ну і остання група – загальна реферація зазвичай вирішує задачу реферації текстів без надання переваги певним словам, зазвичай в залежності від алгоритму видаляються лише певні стоп-слова та значущість слова

визначають за допомогою статистичних показників, семантичного значення тощо. В той час, методи реферації по запитам (*query-focused summarization*) зазвичай сфокусовані тільки на певній тематиці Q, яку користувач задає для формування рефератів виключно певної тематики – це необхідно, наприклад, для реферування текстів, що містять в собі багато різної тематики (наприклад, соціальної та економічної) та для побудови інформативних систем.

Зважаючи на кількість різних методів та підходів, доцільно розглядати найбільш поширені методи[11], а саме:

- класичний метод Едмундсона.
- Метод штучної нейронної мережі.
- Латентно-семантичний аналіз.
- Методи, що базуються на представленні тексту у вигляді графу.

1.2.1 Класичний метод Едмундсона

Метод автоматичного реферування Едмундсона поєднує статистичний метод Г. Луна із позиційним та індикаторним методами. Даний метод характеризує модель лінійних вагових коефіцієнтів[10]:

$$Weight(U) = Location(U) + Cuephrase + Statterm(U) + Addterm(U), \quad (1.1)$$

Статистична важливість текстового блоку $Statterm(U)$ обчислюється як нормована по довжині блоку сума ваг слів або словосполучень, що до нього входить.

Вага $Location(U)$ визначається розташуванням блоку в початковому тексті і залежить від того, де з'являється даний фрагмент: на початку, в середині або в кінці, а також чи використовується він в ключових розділах тексту, наприклад, в заголовку або висновках.

Ключовими фразами (*cue phrases*) є лексичні конструкції-маркери, такі як «підсумовуючи вище сказане», «у даній статті», «за результатами аналізу» та інше. Стоп слова вилучаються із оригінального документа.

Основним плюсом даного методу є його простота з точки зору реалізації, але однозначним мінусом є надмірне скорочення текстового документу[11]

1.2.2 Метод штучної нейронної мережі

Штучна нейронна мережа - це обчислювальна модель, що використовується в інформатиці та інших областях досліджень для вирішення завдань на основі підходів до машинного навчання. Кайхах використовував штучні нейронні мережі як метод екстракції речень для узагальнення новинних статей. Метод поділяється на три етапи: нейромережеве навчання, функціональне злиття і вибір речення.

Етап навчання визначає типи речень, які повинні бути представлені в рефераті документа. Людина робить це і система вивчає схему реферування речень. Після навчання штучної нейронної мережі слід визначити співвідношення між ознаками.

Під час навчання машини розглядаються наступні сім ознак:

a) Параграф після назви документу.

b) Розташування абзацу в документі.

c) Розташування речення в абзаці.

d) Перше речення параграфу.

e) Довжина речення.

f) Кількість тематичних слів у реченні.

g) Кількість заголовків у реченні.

Цей крок складається з двох етапів: 1) видалення незвичайних ознак і 2) усунення ефектів загальних ознак. Тому цей крок узагальнює важливі особливості, які повинні існувати в підсумкових реченнях. Після навчання та

узагальнення мережі ця система може використовуватися для вибору важливих речень для реферування текстового документа[11].

Основним недоліком методу штучної нейронної мережі є його обчислювальна складність, хоча у цього методу є перевага в коригуванні надмірного скорочення[11].

1.2.3 Латентно-семантичний аналіз

Метод латентно-семантичного аналізу (ЛСА) дозволяє виявляти значення слів з урахуванням контексту їх використання шляхом обробки великого обсягу текстів[16].

Модель представлення тексту, що використовується в латентно-семантичному аналізі, багато в чому схожа із сприйняттям тексту людиною. Наприклад, за допомогою цього методу можна оцінити текст на відповідність заданій темі.

В якості вихідної інформації використовується терм-документа матриця. Терм-документа матриця - це математична матриця, що описує частоту термінів, які зустрічаються в колекції документів. Рядки відповідають документам в колекції, а стовпці відповідають термінам. До матриці застосовується сингулярне розкладання. Сингулярне розкладання - це математична операція, розкладають матрицю на 3 складових.

Сингулярне розкладання можна уявити у вигляді формули:

$$A = U \times S \times VT, \qquad (1.2)$$

де A - вихідна матриця, U і VT - ортогональні матриці, а S - діагональна матриця, значення, на діагоналі якої називаються сингулярними коефіцієнтами матриці A. Сингулярне розкладання дозволяє виділити ключові складові вихідної матриці.

Основна ідея ЛСА полягає в тому, що якщо в якості матриці A використовувалася терм-документа матриця, то матриця A *, що містить тільки k перших лінійно незалежних компонент, відображає основну структуру різних

залежностей, присутніх у вихідній матриці. Структура залежностей визначається ваговими функціями термів.

Як правило, вибір k залежить від поставленого завдання і підбирається емпірично[16]. Якщо вибране значення k занадто велике, то метод втрачає свою потужність і наближається за характеристиками до стандартних векторних методів. Занадто мале значення k не дозволяє вловлювати відмінності між схожими термами або документами. Якщо ж необхідно вибирати значення k автоматично, то можна, наприклад, встановити граничне значення сингулярних коефіцієнтів і відкидати всі рядки і стовпці, відповідні сингулярним коефіцієнтам, що не перевищує дане порогове значення.

Схожість між будь-якою комбінацією термів і/або документів найчастіше обчислюють за допомогою скалярного добутку їх векторів, однак на практиці кращий результат дає обчислення схожості за допомогою коефіцієнта кореляції Пірсона. ЛСА відображає документи і окремі слова в так званий «семантичний простір», в якому і проводяться всі подальші порівняння.

При цьому робляться такі припущення:

a) Документ це просто набір слів. Порядок слів у документах ігнорується. Важливо тільки те, скільки разів те чи інше слово зустрічається в документі.

b) Семантичне значення документа визначається набором слів, які, як правило, йдуть разом.

c) Кожне слово має єдине значення. Це, безумовно, сильне спрощення, але саме воно дозволяє вирішити проблеми.

До переваг ЛСА зазвичай відносять екстракцію семантично близьких речень, навіть якщо вони не пов’язані спільними словами – але ЛСА є досить ресурсоємкою групою методів[16].

1.2.4 Методи, що базуються на представленні тексту у вигляді графу

TextRank і LexRank - це алгоритми ранжування на основі графів. Вони працюють на основі PageRank алгоритму. Ребра представляють семантичну

подібність, а вершини – речення. Для побудови ребер найчастіше використовують різні метрики, щоб зрозуміти, які саме вершини пов’язані – наприклад, косинус подібності чи TF-IDF[19]. Ці методи вважають оптимальними[10] для більшості текстів, і тому ми розглянемо один з них (TextRank) так як методи є досить схожими за своєю суттю. Репрезентація тексту за допомогою алгоритму у вигляді графу приводить до двох цікавих результатів. По-перше, підграфи в цьому графі репрезентують підтеми в тексті, по-друге, в центрі графу зазвичай знаходяться найбільш значущі та найбільш пов’язані з іншими речення.

Застосування TextRank до задачі автоматичного реферування текстів представлено в роботі[15]. Алгоритм полягає в наступному:

a) По тексту будується зважений неорієнтований граф, вершини в якому позначають речення тексту. Вагою ребра між двома вершинами є ступінь схожості двох речень, відповідних вершин. Початкова вага розраховується на базі коефіцієнту подібності, який введено останнім у пункті 2.2. та розраховується як кількість спільних слів у реченнях поділена на їх нормовану сумарну довжину.

b) Виходячи з ваг ребер, за допомогою ітераційного процесу кожній вершині присвоюється вага за такою формулою:

$$W(V_i) = (1-d) + d \cdot \sum_{V_j \in Inc(V_i)} \frac{w_{ji}}{\sum_{V_k \in Inc(V_j)} w_{jk}} W(V_j), \qquad (1.3)$$

де V_i, V_j - вершини графа;

$Inc\ (V_i)$ - множина вершин, суміжних з вершиною V_i;

w_{ij} - вага ребра між вершинами V_i, V_j;

d - коефіцієнт загасання, який в даному алгоритмі обирається імперативно, та в роботі[15] дорівнює 0.85;

Ітераційний процес завершується, як тільки ваги вершин перестають змінюватися більш, ніж на 0.0001. (також, обирається емпірично).

с) Після обчислення ваг вершин вони упорядковуються по спадаючому значенню ваги і в реферат включаються речення, що відповідають першим n вершин, де n - бажана кількість речень в рефераті.

TextRank можна використовувати як для реферування одного документа чи для реферації набору документів. Також слід зазначити, що схема для оцінки близькості вершин з використанням чисто статистичних величин, як TF-IDF має свої обмеження – наприклад, цей підхід зазвичай використовує лише частоту того, чи іншого слова, що відкидає певні семантичні особливості тексту. У роботі[15] метрика для визначення ваги ребра також не враховує семантичні чи синтаксичні величини. Звичайно, використання модифікованої функції для обчислення ваги ребра, яка буде враховувати синтаксичні та семантичні особливості тексту значно покращує результати реферату, але також звужує його застосування лише до однієї мови та зазвичай збільшує складність цього алгоритму.

1.3 АНАЛІЗ ВІДОМИХ ПРОГРАМНИХ РІШЕНЬ

На сьогоднішній день відомо багато готових пакетів, що реалізовують алгоритми, зазначені в розділі 1.2, та ведеться багато досліджень, щодо генерування нових методів, пов’язаних з генеруванням реферату, а не екстракції його за допомогою апарату нейронних мереж тощо[14][15].

Розглянемо найвідоміші реалізації – як готові рішення, так і пакети для використання в своїх додатках:

- система AutoSummarize[30], яка вбудована в програмний продукт Microsoft Word. Для кінцевого користувача неможливий вибір методу реферування, можливо лише обрати, наскільки об’єм тексту буде скорочений. Саме з цього можна зробити висновки, що цей застосунок використовує

класичний алгоритм Едмундсона. Звісно, зважаючи на недосконаліть словників, що встановлені та недостатню кількість “cue phrases” для російської та української – система не працює якісно.

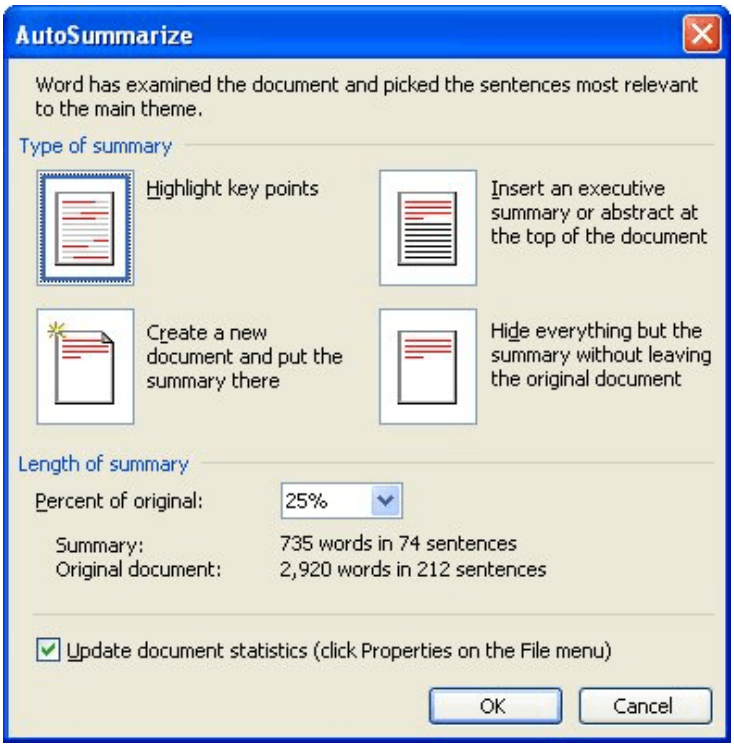


Рисунок 1.1 – Система AutoSummarize

- Веб-застосунок www.textcompactor.com[32] схожий за функціоналом на smmry.com, єдина відмінність – це якість вихідного реферату, що може означати або відсутність семантичного аналізу, або недосконалі словники.

- Веб-застосунок glvrd.ru[33] – реалізовує багато методів обробки даних для оцінки чистоти та читаємості тексту, має досить зручний користувацький інтерфейс та може використовуватись для оцінки якості рефератів з існуючих текстів. З недоліків – відсутність вбудованих методів реферування та відсутність програмного інтерфейсу.

- Пакет с готовими рішеннями для реферування текстів Sumy[34] – може використовуватись розробниками та має велику кількість реалізованих методів – Класичний метод Едмундсона, метод Луна, ЛСА, LexRank та TextRank. З недоліків можна вказати лише на часткову підтримку української мови та на

відсутність підтримки цього пакету для будь-якої мови програмування відмінної від Python.

- Веб-застосунок smmry.com[31] також використовує досить базові алгоритми, засновані на екстракції, але основним недоліком є відсутні словники української та/або російської мов, що унеможливлює використання цього застосунку для створення автоматичних рефератів українською чи російською мовами.

Рисунок 1.2 – Система SMMRY

- Пакет с готовими рішеннями для реферування текстів Sumy[34] – може використовуватись розробниками та має велику кількість реалізованих методів – класичний метод Едмундсона, метод Луна, ЛСА, LexRank та TextRank. З недоліків можна вказати лише на часткову підтримку української мови та на відсутність підтримки цього пакету для будь-якої мови програмування відмінної від Python.

- Пакет с готовими рішеннями для реферування текстів TextRank4ZH[35] – реалізовує алгоритм TextRank для китайської (мандаринської) мови, містить в собі словник і є одним з небагатьох відомих програмних пакетів для китайської

мови. З недоліків можна виділити погану документацію, підтримку тільки одного алгоритму та підтримку однієї мови.

- Морфологічний аналізатор pymorphy2 [20] – рішення, для морфологічного аналізу – тобто, перетворення тексту в вектор грамем, який потрібен для деяких методів реферування текстів. Аналізатор pymorphy2 використовує словник pymorphy2-dicts-uk, що базується на даних з веб-ресурсу languagetool.org[36]. Варто зазначити, що цей пакет був створений для української та російської мови, але безпосередньо методи реферування не імплементовані, що, очевидно, є недоліком, як і відсутність користувацького інтерфейсу.

В роботі були проаналізовані також і інші програмні рішення, але їх за недоліками можна прирівняти до тих рішень, які були названі – або ці рішення не дають можливості напряму обирати алгоритм роботи та мають проблеми з російською та українською мовами – або ці рішення дуже вузькоорієнтовані на певний тип пакету та потребують доопрацювання для адекватного їх використання для вирішення задачі реферування текстових документів російською чи українською мовами.

Для аналізу існуючих застосунків, приведемо порівняльну таблицю

Таблиця 1.1 – Порівняння існуючих програмних застосунків

Застосунки / Критерії	Auto Summarize	Smmry .com	Textcompactor .com	Sumy	Text Rank 4ZH	Pymorphy 2	Glvrd.ru
Наявність інтерфейсу користувача	Так, вбудовано в Microsoft Word	Так, через веб-сторінку	Так, через веб-сторінку	Відсутній	Відсутній	Відсутній	Наявний
Наявність програмного інтерфейсу	Відсутній	Відсутній	Відсутній	Наявний для мови python	Наявний для мови python	Наявний для мови python	Відсутній
Підтримка різних методів реферування	Відсутня	Відсутня	Відсутня	Наявна, велика кількість методів екстракції	Тільки Text Rank	-	-

Формальна підтримка кирилічних мов	Наявна	Відсутня	Відсутня	Наявна	Відсутня	Наявна	Наявна
Наявність документації	Відсутня	Відсутня	Відсутня	Наявна	Відсутня	Наявна	Відсутня
Доступність	Частина програмного пакету Microsoft Word	Доступно як веб-ресурс	Доступно як веб-ресурс	Потребує реалізації	Потребує реалізації	Потребує реалізації	Доступно як веб-ресурс

Продовження таблиці 1.1

Якість реферування російською	Погана	Відсутня	Відсутня	Добра	Відсутня	-	-
Якість реферування українською	Погана	Відсутня	Відсутня	Погана	Відсутня	-	-
Врахування семантичних особливостей мови	Погане	Наявне	Відсутнє	Наявне (для деяких методів)	Відсутнє	Найкраще з аналогів	Добре

ВИСНОВКИ ДО РОЗДІЛУ 1

В розділі були наведені основні відомості про методи обробки текстів, а саме методи автоматичного реферування та анотування текстів, розглянуті більш детально існуючі програмні застосунки та зроблені висновки щодо ефективності цих застосунків при використанні їх для вирішення задачі автоматичного реферування текстових документів написаних українською та російськими мовами. Також були наведені теоретичні відомості про критерії та оцінки, які найчастіше використовують для оцінки якості отриманого реферату – причому, як використовуючи зразковий реферат, так і використовуючи автоматичний метод, який не потребує цього реферату.

Зважаючи на існуючі математичні методи вирішення проблеми реферування, найбільш доцільно буде використовувати існуючий апарат методів екстракції – як найбільш досліджених та імплементованих. Зважаючи на простоту цих методів, найбільш оптимальним буде вибір гібридного методу – використання латентно-семантичного аналізу для розрахування ваги речень

та використання методів, що використовують представлення тексту у вигляді графу.

2 МАТЕМАТИЧНІ МОДЕЛІ ЗАДАЧІ РЕФЕРУВАННЯ

Для вирішення задачі реферування, необхідно визначитись з її математичною формою. Задача реферування з математичної точки зору потребує певного визначення поняття тексту, математичної моделі тексту та формального визначення реферату, а також визначення допустимих операцій з текстом. На даний момент існує досить багато різних математичних моделей тексту, у кожної є свої переваги та недоліки. Доцільно розглянути деякі з моделей та оцінити, наскільки вони підходять саме до задачі автоматичної реферації.

2.1 МАТЕМАТИЧНЕ ПРЕДСТАВЛЕННЯ ТЕКСТУ

2.1.1 Модель 1-hot encoding

Модель 1-hot encoding[21], або модель унітарного кодування є досить базовою моделлю, яка часто використовується у багатьох задачах як обробки людських мов – так і у багатьох задачах машинного навчання для репрезентації знання. Наведемо приклад його роботи. Нехай у нас є текст з кількістю унікальних слів n – тоді для кожного окремого слова ми можемо обрати унікальний вектор розміру n – який і буде представляти це слово.

Таблиця 2.1 – Представлення тексту унарним кодуванням

Слово	Я	вчусь	на	шостому	курсі
Код	00001	00010	00100	01000	1000

Як бачимо, модель унітарного кодування не враховує ні морфологічні особливості, ні зв’язки між словами та реченнями, ні семантичні особливості тексту. Тому, використовуючи модель 1-hot encoding неможливо визначити близькість слів чи словосполучень для створення реферату та, навіть, не можна

математично визначити саме поняття реферату. Тому необхідно розглянути інші моделі, які можна використати.

2.1.2 Модель представлення тексту у вигляді n-грам

Від досить простої моделі унітарного кодування розглянемо більш складну – модель n-грам[22]. Це статистична модель, що працює з досить великими початковими даними, яка визначає будь-яку послідовність слів h довжиною в n як n-граму та для слова чи послідовності слів s визначає ймовірність:

$$P(h \mid s) = \frac{C(h+s)}{C(h)}, \qquad (2.1)$$

де $C(h+s)$ – це кількість зустрічі послідовностей слів h одразу після якого йде у текстах s та $C(h)$ - це загальна кількість зустрічі послідовностей h у текстах.

Звичайно, розглядають біграми (2-грами), 3-грами і так далі. Моделі n-грам, як і будь-які статистичні моделі дуже сильно залежать від навчальної вибірки та дуже погано працюють з текстами, в яких є невокабулярні слова.

Вирішення задачі реферування при використанні моделі n-грам можливе, особливо якщо задача обрахування більшості n-грам вже вирішена, і якщо для абсолютно усіх слів у тексті ваги обраховані та нормовані. При таких умовах можна задати функцію близькості тексту, але майже неможливо буде обрахувати якість реферату та задати критерії якості. Модель n-грам часто використовується для вирішення задач автоматичного закінчення тексту, наприклад в середовищах розробки саме використовуючи n-грами система прогнозує, який код буде далі написаний. Однак, для вирішення та постановки задачі реферування, ми розглянемо інші методи.

2.1.3 Модель Bag-Of-Words

Модель Bag-Of-Words (або модель «мультимножини слів») є досить простою моделлю, яка побудована за принципом частоти того чи іншого слова[23]. Не дивлячись на простоту цієї моделі, її можна використовувати для вирішення задачі ранжування та порівняння речень у загальному алгоритмі. Для початку, ідентифікуємо цю модель на прикладі цього тексту: «Я вчусь на шостому курсі університету. В університеті досить цікаво, я хочу далі навчатись».

Для даного тексту ми будуємо «мультимножину» (іноземний термін – superset) зважаючи на частоту зустрічаємості того чи іншого слова в даному тексті:

Таблиця 2.2 – Приклад використання моделі Bag-Of-Words

Кількість	Слово
2	Я
1	вчусь
1	на
1	шостому
1	курсі
1	університету
1	В
1	університеті
1	досить
1	цікаво
1	хочу
1	далі
1	навчатись

З даного прикладу, можна вже зрозуміти потенційні недоліки цього методу репрезентації тексту – ніяк не враховуються морфологічні особливості, тому одне і те саме слово університет у двох відмінках рахується окремо для кожного відмінку.

Другим очевидним недоліком є ігнорування порядку слів – очевидно, для побудови тексту порядок слів є важливим – тому абстрактні методи (генерування реферату) дуже погано працюють з таким форматом репрезентації тексту. Третім недоліком є ігнорування синтаксичних символів, що породжує трішки вдосконалену модель яка називається Bag-Of-Terms або «мультимножина термів».

На відміну від Bag-Of-Words, основним елементом є не слово, а терм – символьний вираз формальної моделі мови. Тому синтаксичні символи також можуть розглядатись як елементи мультимножини.

З точки зору задачі реферування, модель Bag-Of-Words може бути використана щоб виділити основі елементи тексту в екстрактивних методах – зокрема, для ранжування речень зважаючи на їх вагу, яка розрахована з суми ваг слів, які зустрічаються у тексті. Детальніше про показники, які можна обраховувати у рамках частотної моделі тексту буде вказано у пункті 2.3. Зазначимо, що зважаючи на недоліки екстрактивні методи не можуть давати добрі результати використовуючи тільки Bag-Of-Words, тому є сенс розглянуті й інші математичні моделі тексту.

2.1.4 Векторна модель тексту

Якщо шукати матеріали, щодо математичної моделі тексту найчастіше можна зустріти поняття про векторну модель тексту. Якщо розглядати текст у векторній моделі, то йому надана така дефініція: Текст – це невпорядкована множина термів, де терми – символьний вираз формальної моделі мови (це слово, синтаксичний символ, число тощо). Вектором тексту є представлення

тексту у векторному просторі, де кожному терму (навіть якщо він відсутній у конкретному документі) надається певна вага (за допомогою частотного аналізу тощо)[24]. Завдяки цьому можна зберегти однаковий розмір векторів для різних текстів. Наприклад, нехай у нас є текст з розмірністю у n різних термів, який має N різних речень. Тоді для побудови вектору речення необхідно обрахувати:

$$d_m = < w_{1m}, \dots, w_{nm} >,$$

де d_m – вектор конкретного речення, а w_{im} – вага і-того терму у цьому реченні. Як відомо, існує великий математичний апарат для порівняння векторів, а вибір функції для обчислення ваги може бути різним – від найпростішого бінарного індексу (чи містить цей терм це речення – чи ні) до більш складного ітеративного процесу з використанням нейронних мереж.

Для деяких задач обробки природніх мов необхідно обраховувати так звані words embeddings або вкладення слів – це значення близькості двох окремих слів. Нехай у нас є класифікатор який відповідає на досить просте запитання – наскільки слово v близьке до слова w? Якщо не розглядати частотні показники, то вирішення багатьох задач, включаючи реферування, так чи інакше зводиться до побудови методу порівняння слів чи речень. Задача, у якій ми шукаємо вектори подібності одного слова до кожного іншого, вирішується двома моделями, що походять від векторної – word2vec та GloVe. Зважаючи на нашу задачу, ми будемо розглядати векторну модель тексту та модель тексту на графі, так як для вирішення задачі екстенсивного реферування використання моделей word2vec та GloVe є надлишковою роботою.

2.1.5 Модель тексту у вигляді графу

У розділі 2.1.4. ми розглянули вектору модель тексту. Одним з недоліків цієї моделі є велика залежність від функції ваги слова у реченні – ця функція може не враховувати структурну особливість зберігання інформації у тексті. Однією з дуже близьких альтернативних моделей є модель графічної

репрезентації тексту (Graph-based text model)[25], де формально визначаються дві множини – вершин та ребер.

Існує декілька різних видів термів, що використовуються як вершини у текстовому вигляді – це можуть бути леми, стеми, слова тощо, але найчастіше у вигляді ребер обирають розміщення слів всередині речень, текстів. Ця модель дозволяє обробляти синоніми, гіпоніми майже без втрат.

Розглянемо принцип побудови графа з тексту. Нехай у нас є речення: «Я люблю КПІ». Тоді його графічним представленням буде орієнтований граф:

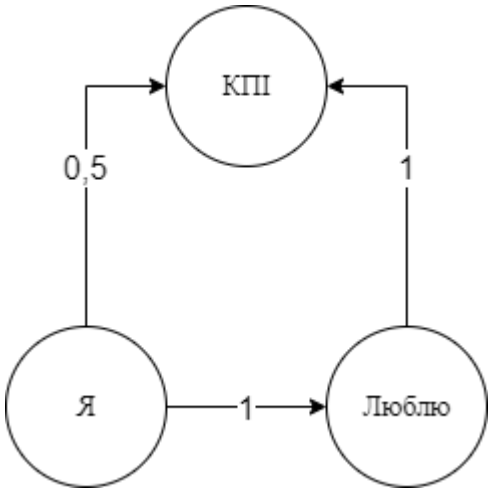


Рисунок 2.1 – Графічна модель тексту

А сам граф можна описати так:

$$G = < V = \{\text{Я}, \text{Люблю}, \text{КПІ}\}, E = \{< \text{Я}, \text{Люблю}, 1 >, < \text{Люблю}, \text{КПІ}, 1 >, < \text{Я}, \text{КПІ}, 0.5 >\}$$

У цьому графі ваги нормовані, але в цілому є досить багато різних способів визначити вагу того, чи іншого ребра. Ці значення розраховуються як з синтаксичної близості, так і через кількісні параметри. Зважаючи на те, що і векторна модель і графічна модель при розрахунках і для вирішення задачі ранжування речень (яка необхідна для вирішення задачі автоматичного реферування) опираються на вагу слова у векторі чи на вагу ребра у графі,

доцільно буде розглянути різні функції, які можна використовувати для вирішення цієї задачі.

Зазначимо, що в задачі реферування доцільно вважати речення векторами слів, а сам граф будувати з множини речень як вершин та ребер, як зв’язків між цими ребрами.

2.2 МЕТОДИ ВИЗНАЧЕННЯ ВАГИ СЛІВ ТА РЕЧЕНЬ

Як ми зрозуміли з минулого параграфу, найбільш доцільними моделями які можна використовувати у задачі автоматичного реферування є векторні та графічні моделі, які залежать від вибору величин на роль функції ваги. Розглянемо найпопулярніші методи більш детально.

Однією з найбільших і найпримітивніших (хоча і досить ефективних) груп методів є кількісні характеристики:

- бінарний метод: Досить проста функція, 1 якщо речення включає в себе терм, і 0 якщо не включає (Математична функція індикатора);
- кількість входження термів у текст;
- частота вхождення терму у текст (TF – Term Frequency);
- логарифм від частоти вхождення терму у текст (LOGTF);
- обернена частота вхождення (IDF – Inversed Document Frequency).

Розглянемо детальніше групу TF-IDF. TF вираховується як відношення кількості входжень слова до загальної кількості слів у тексті. При загальній простоті характеристики, вона забезпечує прийнятний результат для методів інформаційного пошуку та класифікації (пропонує неспіврозмірність оцінки для текстів різної довжини - недооцінює довгі речення, так як вони є більш складними і середня частота слів в тексті нижче). LOGTF це вага вхідного в текст речення визначається як 1 + log (TF), де TF - частота терміну. Використання логарифмічної шкали дозволяє зробити модель більш стійкою до переоцінки текстів різного обсягу.

Щодо IDF, то вона розраховується по формулі:

$$IDF(t, D) = \log \frac{|D|}{|\{d_i \in D \mid t \in d_i\}|}, \qquad (2.2)$$

Тобто, це логарифм від загальної кількості речень до кількості речень, що містять певний терм.

Для більшості задач оптимальним кількісним параметром є добуток величин TF та IDF, в особливості для задачі реферування. Більше того, знаючи цей добуток можна вирахувати так званий косинус подібності.

Косинус подібності розраховується як косинус перетину між векторними поданнями TF-IDF між сукупністю речень, що входять до тексту та суцільним текстом.

$$\cos\theta = \frac{TFIDF_{input} \cdot TFIDF_{full}}{|TFIDF_{input}||TFIDF_{full}|}, \qquad (2.3)$$

Така кількість величин, які є де-факто величинами побудованими за допомогою тільки кількісних параметрів, пов’язаних з кількістю слова в реченні чи частотою слова у всьому тексті. Звісно, не тільки таким чином можна оцінити схожість тексту. Розглянемо і деякі інші кількісні міри подібності.

Розглянемо наприклад таку величину як позиційний коефіцієнт. Цей коефіцієнт був введений у роботі[26] та полягає у наступному майже емпіричному підході, що найбільш важливий текст знаходиться на самому початку тексту – нехай N – кількість речень, тоді для речення і оцінка буде:

$$S_i = 1 - \frac{i - 1}{N}, \qquad (2.4)$$

Аналогічно можна розглянути також й інші параметри для оцінки ваги. Зазначимо, що для більшості алгоритмів достатньо і цих параметрів. Для алгоритмів, що використовують репрезентацію тексту у вигляді графу, часто використовують так званий коефіцієнт подібності, який розраховується для двох речень S_i та S_j по формулі

$$Sim(S_i, S_j) = \frac{|W_k|W_k \in S_i \,\&\, W_k \in S_j|}{\log(|S_i|) + \log(|S_j|)}, \qquad (2.5)$$

Тобто, розраховується кількість загальних термів в двох реченнях що ділиться на суму логарифмів довжини цих речень (щоб нівелювати різну довжину речень).

Зважаючи на кількість різних параметрів та математичні моделі для репрезентації тексту, можна вводити математичне формулювання задачі реферування обираючи той, чи інший коефіцієнт.

2.3 ЗАДАЧА РЕФЕРУВАННЯ ІЗ МАТЕМАТИЧНОЇ ТОЧКИ ЗОРУ

Задача реферування може мати різні формулювання в залежності від обраних мір подібності тексту та, навіть, від обраної математичної моделі репрезентації текстів. В даній роботі ми будемо використовувати векторну модель та модель на графі. Також це формулювання справедливе для методів екстракції а не на методах генерації рефератів.

Формулювання задачі реферування для векторної моделі тексту:

Нехай T – це весь текст до реферату, який є множиною речень S_i, які являють собою вектори. Тоді множину R – називають рефератом, якщо виконуються такі умови:

- $|R| < |T|, |R| > 0$
- R – може бути як підмножиною T, так і не входити в T.
- Косинус подібності $\cos\theta \rightarrow 1$.

Пояснення – реферат, ненульова множина векторів термів (речень), яка максимально схожа за змістом до початкового тексту.

Формулювання задачі реферування для моделі тексту на графі:

Нехай T – це граф, в якому множина вершин це речення (вектори термів) S_i, які пов'язані між собою зв'язками e_{ij}. Тоді граф R – називають рефератом, якщо для нього виконуються такі умови:

- |R| > 0, множина вершин R ∈ множині вершин T.

- Для фіксованої кількості $S_i \in R, \forall S_j, j \neq i, e_{ij} \rightarrow max$.

Тобто ми шукаємо найбільш зв'язний підграф який входить до початкового графу.

2.4 ПОСТАНОВКА ЗАДАЧІ

В даному розділі ми розглянули математичні методи, які використовуються для вирішення задачі реферації методом екстракції – тобто виділення реферату з готового тексту. Зважаючи на математичне формулювання задачі реферування, розглянуті моделі та методи, доцільно надати вимоги до алгоритму реалізації та програмного забезпечення, що буде його реалізовувати.

Наведемо вимоги до реалізації:

- створений програмний застосунок має реалізовувати хоча б один з екстрактивних методів реферації для української та російської мов.

- Створений програмний застосунок має реалізовувати алгоритм підготовки тексту, алгоритм реферування та оцінки результату за допомогою математичного апарату з п. 2.2. чи програмних пакетів оцінки.

- Результат роботи програмного застосунку має бути рефератом з математичної точки зору.

- Створений програмний застосунок має працювати з текстовими документами будь-якої довжини, написаними українською або російською мовою.

ВИСНОВКИ ДО РОЗДІЛУ 2

В розділі були розглянуті основні математичні методи представлення тексут, а саме 1-hot word enconding, Bag-Of-Words, Bag-Of-Terms, векторне представлення тексту та представлення тексту у вигляді графу. Зважаючи на представлені моделі тексту, для деяких з них було розглянуто приклади вагових функцій, які, як було показано у розділі 1.2 важливі для правильної роботи алгоритмів екстрактивного реферування. Розглянуті такі вагові показники, як частотні (зокрема TF-IDF, косинус подібності, бінарний метод) та інші міри подібності. Зважаючи на показники, було формалізовано задачу реферування з математичної точки зору для двох моделей представлення тексту – векторної та графічної.

Як результат, зважаючи на математичній апарат представлений у першому та другому розділі, та формалізацію задачі реферування, була поставлена задача для магістерської дисертації, з наведеними вимогами до алгоритму та його програмної реалізації.

3 АЛГОРИТМ АВТОМАТИЧНОГО РЕФЕРУВАННЯ

3.1 ЗАГАЛЬНИЙ АЛГОРИТМ ЕКСТРАКТИВНОГО РЕФЕРУВАННЯ

Для майже усіх алгоритмів екстрактивного реферування основний підхід заключається у трьох послідовних етапах обробки тексту та ітерації по реченням. Відмінність наступає тільки у кількості підготовки на початковому етапі, методам визначення вагових коефіцієнтів та від самого представлення тексту. Приведені далі методи TextRank та LSA мають відмінність саме у представленні тексту та ітераціях по ранжуванню необхідних речень.

Псевдокод, який реалізує цей загальний підход для вирішення задачі реферування:

```
function summarize (var text, var language, var size):
    var preparedText;
    if (containsStopWords (text, language)):
        preparedText = removeStopWords (text, language)
    if (containsCuePhrases (preparedText, language)):
        preparedText = removeCuePhrases (preparedText, language)
    preparedText = tokenize(preparedText)
    preparedText = stemming(preparedText)
    for (sentence in preparedText):
        weightMatrix[sentence] = calculateWeight (sentence, preparedText)
    for (sentence in preparedText):
        if (preparedText.length <= size): return preparedText
        else:
            relalculateWeights(preparedText)
            if (lowest(weightMatrix[sentence], preparedText):
                preparedText = removeSentence (preparedText, sentence)
```

Наведемо блок-схему загального алгоритму екстрактивного реферування:

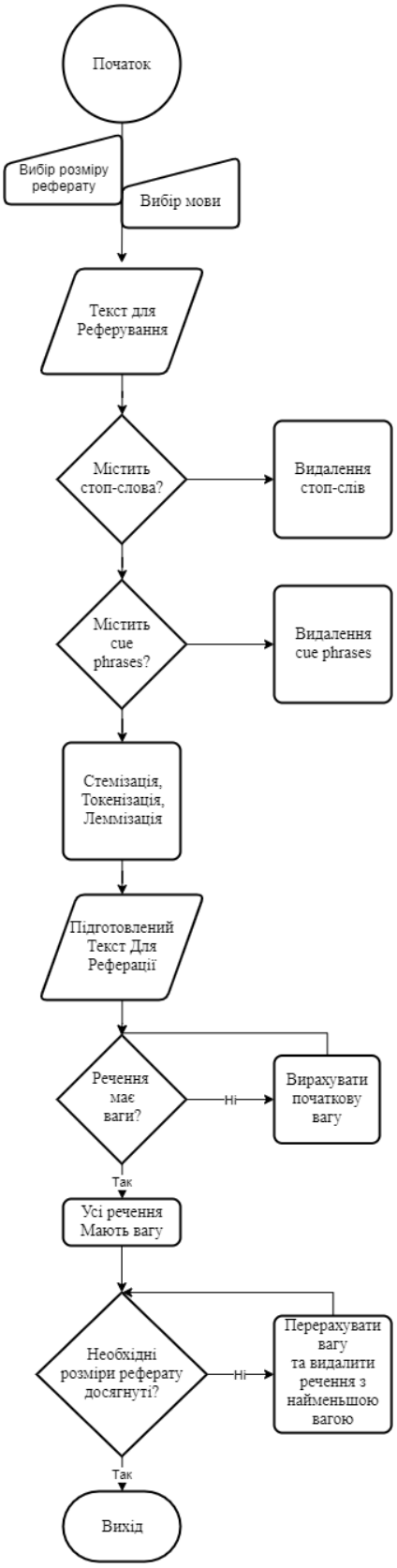


Рисунок 3.1 – Блок Схема загального алгоритму екстрактивного реферування

Як бачимо, загальна схема алгоритмів екстракції надає нам можливості для зміни лише в останніх двох ітераціях по реченням. Звісно, існуючі рішення можна покращити навіть без зміни алгоритму чи функції вибору ваги того чи іншого речення просто покращив результати на етапі стемінгу та токенізації. Також нагадаємо, що стемінг – це приведення слова до форми стеми (відкидання суфіксу та закінчення), лематизація (приведення слова до його нормальної форми).

3.2 АЛГОРИТМ TEXTRANK

Даний алгоритм є представником підходів, заснованих на графах. Його загальний опис надано у пункті 1.2.4, тому у цьому розділі ми наведемо псевдокод та блок-схему роботи цього алгоритму. Зазначимо, що зважаючи на блок-схему та псевдокод наведений у пункті 3.1 є сенс виключно у репрезентації частини, що безпосередньо продукує реферат з вже підготовленого коду.

Псевдокод алгоритму:

```
function textrank (var preparedText, var size, var threshold, var d):
    var weights
    for (i in preparedText):
        for (j in preparedText):
            weights[i][j] = totalCommonWords (i, j) / (size (i) + size (j))
    do:
        var previousWeights = weights
        for (i in preparedText):
            for (j in preparedText):
                weights[i][j] = calculateWeight (d, i, j)

    while (changedBelowThreshold (previousWeights, weights, threshold))
```

```
for (sentence in preparedText):
        if (lowest_scored (size, preparedText, weights)):
                preparedText = removeSentence (preparedText, sentence)
return preparedText
```

Наведемо також блок-схему алгоритму TextRank:

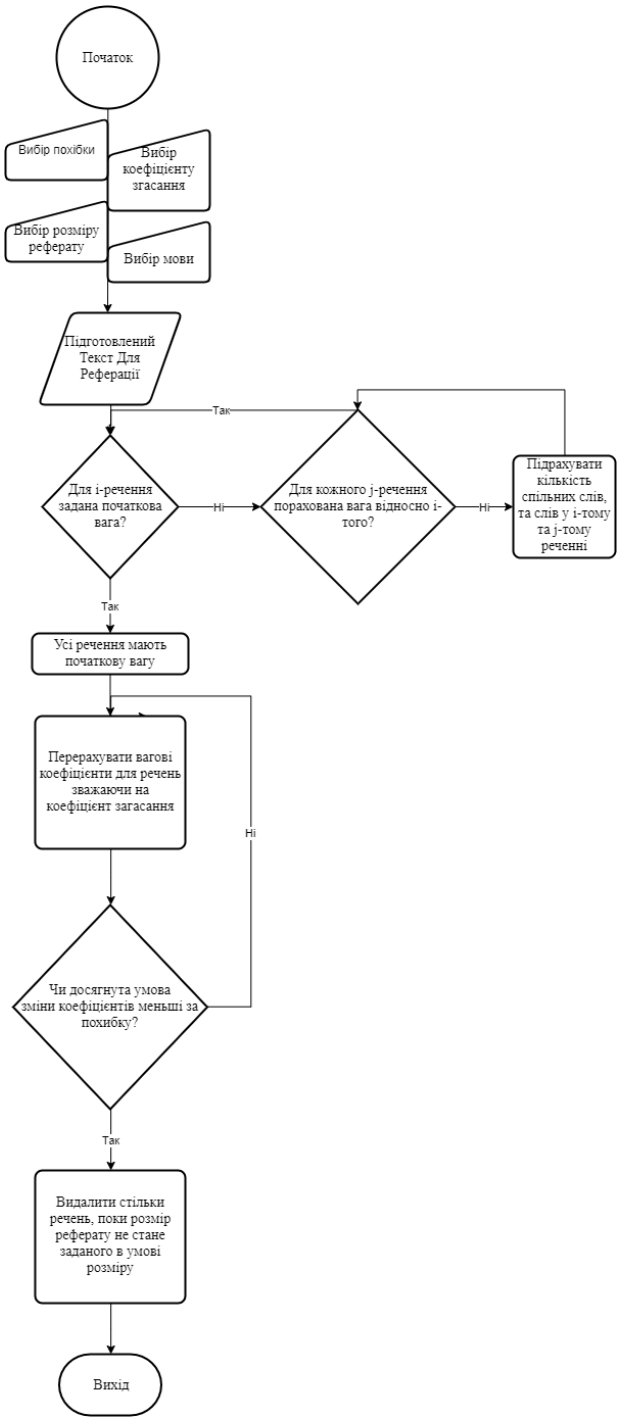


Рисунок 3.2 – Блок-схема алгоритму TextRank

3.3 АЛГОРИТМ LSA

Даний алгоритм заснований на прихованому семантичному аналізі (Latent Semantic Analysis), що дозволяє отримати неявне уявлення текстової семантики на основі спільної зустрічальності слів. Вперше використання цього методу для реферування одиночних документів було запропоновано в [16]. Він полягає в наступному:

a) Нехай A - матриця терм-речень, отримана за вхідним документом. Її розмір дорівнює $n \times m$, де n - кількість термів в документі, m - кількість речень. Елемент a_{ij} цієї матриці дорівнює частоті терма i в тексті, якщо цей терм зустрічається в реченні j і 0 в іншому випадку.

b) До отриманої матриці застосовується сингулярне розкладання:

$$A = U\Sigma V^T, \qquad (3.1)$$

де $U = [u_{ij}]$ - ортонормована матриця розміру $n \times m$, $\Sigma = diag\ (\sigma_1, \sigma_2,..., \sigma_m)$ - діагональна матриця, $V = [v_{ij}]$ - ортонормована матриця розміру $m \times m$. Якщо $rank\ (A) = r$, то виконується:

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq \sigma_{r+1} = \sigma_m = \cdots = 0$$

З точки зору семантики сингулярне розкладання матриці A інтерпретується як розбиття вхідного документа на r концепцій (тем). Кожен елемент v_{ij} матриці V відображає ступінь інформативності пропозиції j по темі i. При цьому значення σ_i матриці Σ відображає ступінь важливості теми i в вихідному документі.

c) Кожному реченню s_k вихідного документа присвоюється вага за формулою:

$$s_k = \sqrt{\sum_{i=1}^{m} v_{ik}^2 \cdot \sigma_i^2}, \qquad (3.2)$$

Тобто більшої ваги набувають речення, найбільш інформативні за однією з тем документа, при цьому враховується і ступінь важливості концепції в документі.

d) Значення ваг речень упорядковуються за спаданням, і в реферат включаються пропозиції, що відповідають першим l значенням, де l - бажана кількість речень в рефераті.

Зважаючи на детальний опис алгоритму, надамо блок-схему для нього:

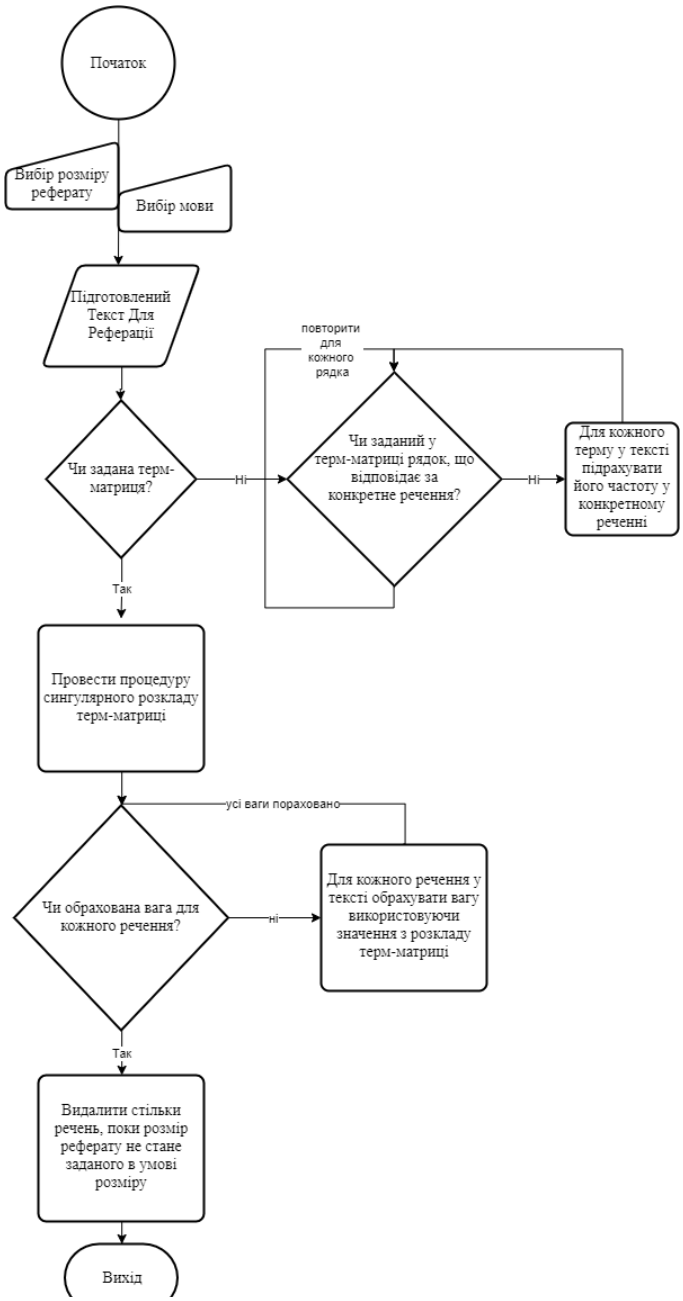


Рисунок 3.3 – Блок-схема алгоритму латентно-семантичного аналізу

3.4 ЗАПРОПОНОВАНИЙ АЛГОРИТМ

Розглянувши загальній підхід для вирішення задачі реферування екстрактивними методами та більш детальний метод з використанням алгоритмів TextRank та LSA, можна запропонувати загальний алгоритм, який вмістить у собі варіативність підходу LSA та TextRank щодо розрахунків вагових коефіцієнтів, містить в собі усі необхідні перетворення для покращення якості та має менше недоліків за LSA та TextRank.

Два основних концепти будуть використовуватись для розробки цього концепту – по-перше, для лематизації буде використовуватись сильний морфологічний аналізатор, який працює зі словником OpenCorpora[36] pymorphy2[20]. По-друге, під час виконання операції реферування, одночасно будуть працювати алгоритми TextRank та LSA, але наприкінці буде видаватись тільки той результат, що є кращим за показником косинуса подібності – як одного з незалежних показників, що використовується для оцінки якості роботи алгоритму реферування.

Наведемо псевдокод який буде реалізовувати наш алгоритм:

```
function summarize_special (var text, var language, var size, var preferrableAlgo):
    var preparedText;
    var default_coef = 0.85
    var default_epsilon = 0.001
    if (containsStopWords (text, language)):
        preparedText = pymorhpy2.removeStopWords (text, language)
    if (containsCuePhrases (preparedText, language)):
        preparedText = pymorhpy2.removeCuePhrases (preparedText, language)
    preparedText = pymorhpy2.lemmatize(preparedText)
    if (preferrableAlgo == null):
        lsaText = lsa (text, size)
        textRankText = textRank (text, size, default_coef, default_epsilon)
        preparedText = biggerCosine (lsaText, textRankText)
    else if (preferrableAlgo == 'LSA'):
        preparedText = lsaText (text, size)
    else:
        preparedText = textRank (text, size, default_coef, default_epsilon)
    return preparedText
```

Надамо трішки контексту для роботи цього алгоритму. Звісно, зважаючи на критерії оцінки для малих текстів з одним змістом великого сенсу

порівнювати результати роботи не має – вони скоріше за все будуть ідентичними. Але для текстів з декількома центрами тяжіння (тобто в тексті є багато різних підтем) є потенціал для росту. Також не слід недооцінювати програмну бібліотеку pymorphy2[20] – саме лематизація слів (морфологічний аналізатор вміє працювати як із словниковими словами, так і з несловниковими) у східнослов'янських мовах може значно покращити якість екстрактивного реферату, так як це значно покращить початкові частотні показники. Нагадаю, що для задачі реферування при використанні векторної чи графічної моделі тексту слова «університет» та «університету» вважаються різними, і тому частотні показники можуть різко підскочити для якогось неврахованого cue word, як, наприклад, слово «наприклад».

Програмна імплементація даного алгоритму буде включати в себе як імплементацію LSA, так і імплементацію TextRank. Тому детальна блок-схема для цих алгоритмів потрібна.

Слід зазначити, що для запропонованого алгоритму блок-схема надана у вигляді графічного додатку до диплому у повному вигляді через велику кількість дій у самому алгоритмі. Але для структурної цілістності, надамо блок-схему:

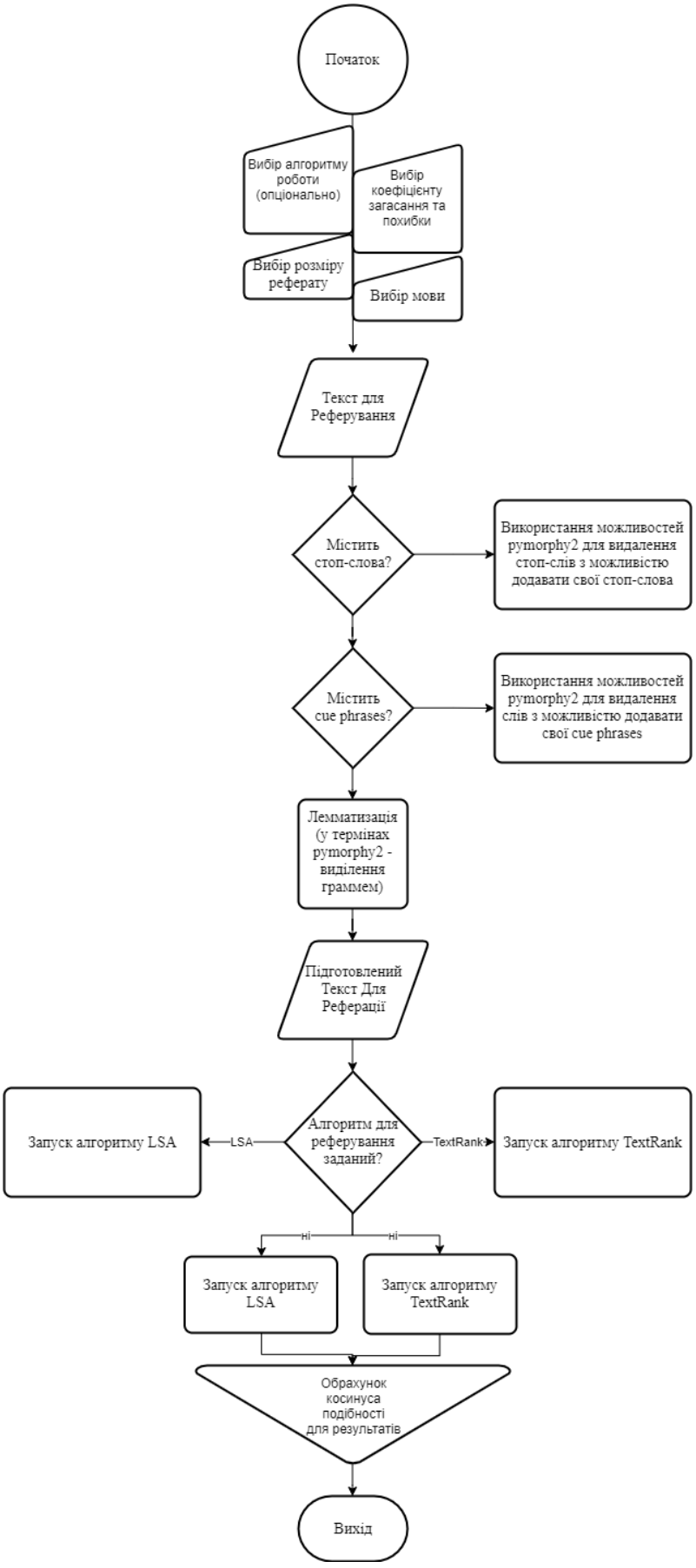


Рисунок 3.4 – Блок-схема запропонований алгоритму

ВИСНОВКИ ДО РОЗДІЛУ 3

Цей розділ було присвячено більш детальному розгляду обраних екстрактивних методів та їх алгоритмів роботи, було розглянуто загальну алгоритмічну форму екстрактивних алгоритмів, розглянуто два найбільш популярних метода екстрактивного реферування – TextRank та метод латентно-семантичного аналізу. У ході порівняння роботи алгоритму та зважаючи на особливості роботи двох алгоритмів, було запропоновано власний алгоритм, який фактично програє у швидкодії через використання апарату одночасно обох методів, але потенційно може давати кращі результати роботи.

Запропонований метод повністю задовільняє формулювання постановки задачі, так як використовує кількістні ваги, використовує декілька методів екстрактивного реферування та виконує підготовку тексту через видалення стоп-слів, лематизацію та інші трансформації перед безпосередньо реферуванням.

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 ВИМОГИ ДО РОЗРОБЛЮВАНОГО ПЗ

Зважаючи на завдання цієї роботи та на аналіз вже існуючого програмного забезпечення, для досягнення позитивного результату, необхідно:

- розробити програмну бібліотеку з підтримкою хоча б однієї мови програмування та хоча б однієї операційної системи. Бажано мати незалежність від платформи, але так як це програмна бібліотека, достатньо мати підтримку найпопулярнішої операційної системи;
- імплементувати запропонований алгоритм та усі необхідні для цього процедури, а саме: алгоритм латентно-семантичного аналізу, алгоритм TextRank, імплементація обрахунків TF-IDF, векторну модель тексту, методи, необхідні для обрахунку косинусу подібності;
- протестувати роботу бібліотеки на текстах різного змісту та авторства. Використати різні підходи для оцінки якості – емпіричний, автоматичний та використати інше програмне забезпечення для порівняння.

Однак, так як ми вирішуємо досить базову задачу реферації текстів нам не потрібно використовувати багато різного функціоналу як, наприклад:

- використовувати бази даних для зберігання якихось даних. В нашому випадку сенс використовувати бази даних виникає лише, наприклад, при побудові сервісу для зберігання історичних даних. Стоп-слова та cue phrases зазвичай не потребують окремих баз даних для зберігання, їх можна зберігати використовуючи прості текстові файли, чи використовуючи формат .json чи .xml;
- імплементувати алгоритми машинного навчання для коректної роботи алгоритмів чи застосунку. В нашому випадку екстрактивні алгоритми, що ми

використовуємо не потребують розгортання штучних нейтронних мереж чи навчання класифікатора;

- будувати інтерфейс користувача, так як для програмної бібліотеки буде достатньо мати документацію та керівництво користувача.

Отже, задача побудови програмного забезпечення є досить простою, але, щоб описати архітектуру, необхідно задати декілька UML-діаграм.

4.2 АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Архітектура розроблюваного застосунку потребує визначення чіткої структури, яка так чи інакше буде відповідати кращим принципам програмування – система має бути відкритою для розширення та закритою для змін, процес міграції та залежностей має вирішуватись просто. Звісно, для того, щоб адекватно описати архітектуру програмного застосунка потрібно надати такі UML-діаграми:

- діаграму класів;
- діаграму послідовностей;
- діаграму розгортання.

Звісно, для цієї розробки також важливо виділити основні архітектурні рішення, які були прийняті, зокрема вибір технологій, операційної системи тощо. У якості мови розробки була обрана мова python[37] версії 3.6, так як саме для цієї мови вже розроблена програмна бібліотека - морфологізатор для російської та української мови pymorphy2[20]. Тому вибір іншої мови програмування є недоцільним, оскільки тоді потрібно було би налаштовувати два набори середовищ, що в еру контейнерів не є великою проблемою, але є недоліком. До того ж, мова python досить проста, популярна та досить проста для використання на майже усіх операційних системах.

У якості операційної системи була обрана система Microsoft Windows 10, але зважаючи на можливості середовища Conda[38] версії 5+ - бібліотека функціонує майже не використовуючи методи операційної системи.

Наведемо тепер діаграму класів:

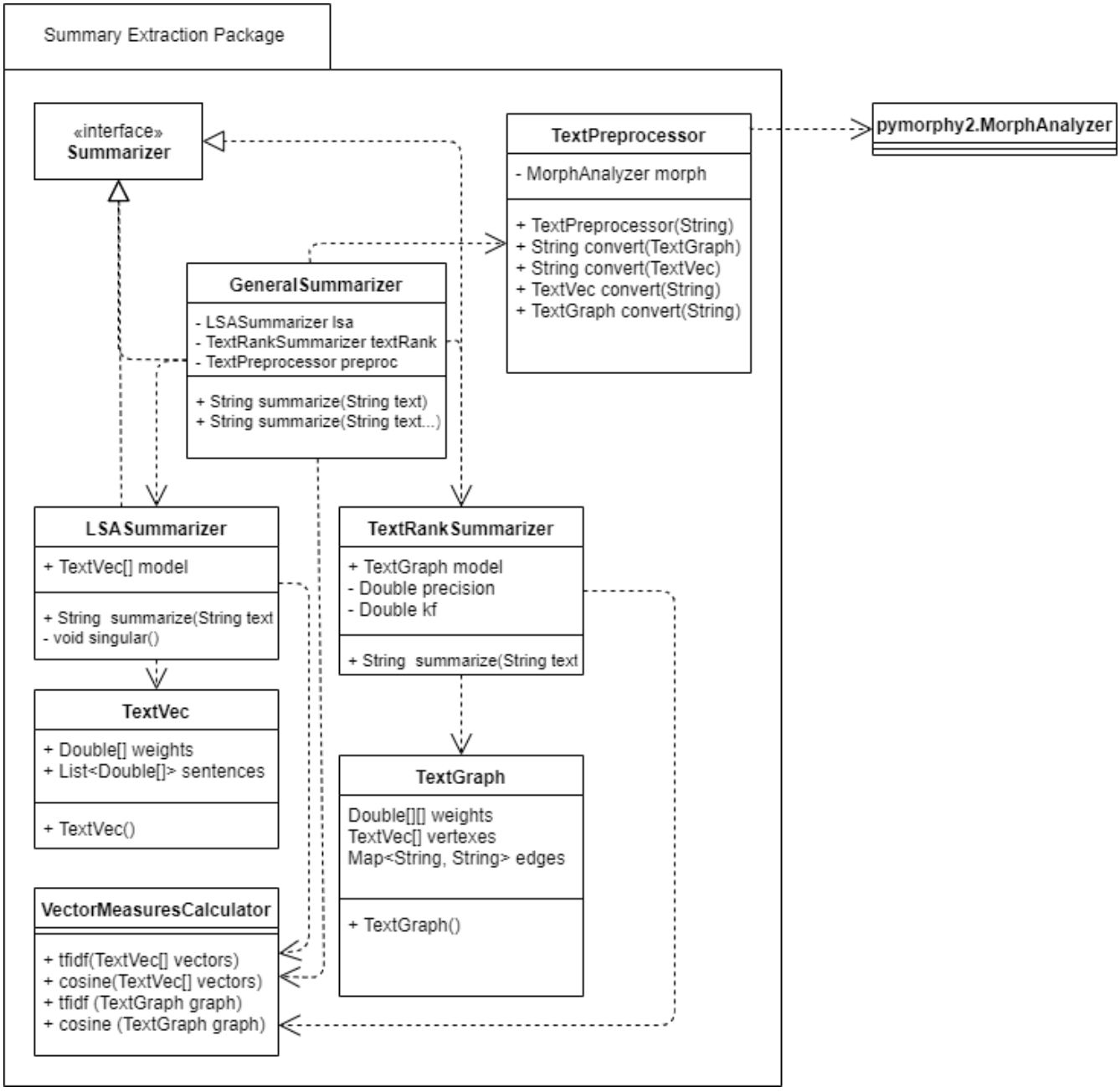


Рисунок 4.1 – UML-діаграма класів

Коментар до діаграми класів – як бачимо, система має 8 публічних класів, три класи, що імплементують інтерфейс екстрактора рефератів – а саме інтерфейс Summarizer. За допомогою об’єктно-орієнтованого підходу, ми виділяємо базовий інтерфейс не тільки для імплементованих класів, що реалізують LSA, TextRank та власний алгоритм, а і для майбутніх імплементацій.

Також виділена модель (класи TextVec там TextGraph) та окремий код для роботи з моделлю – VectorMeasuresCalculator та TextPreprocessor. Також весь пакет зв’язується з пакетом pymorphy2 тільки через клас TextPreprocessor. Для користувачів пакету потрібно буде використовувати виключно три класи сумаризатора та текст процесор, так як потрібно його перетворити на класи моделі. Слід зазначити, що в діаграмі навмисно не вказані юніт-тести, так як ці класи потрібні виключно для підтримки якості і не є логічними елементами у системі.

Далі наведемо діаграму розгортання цього рішення

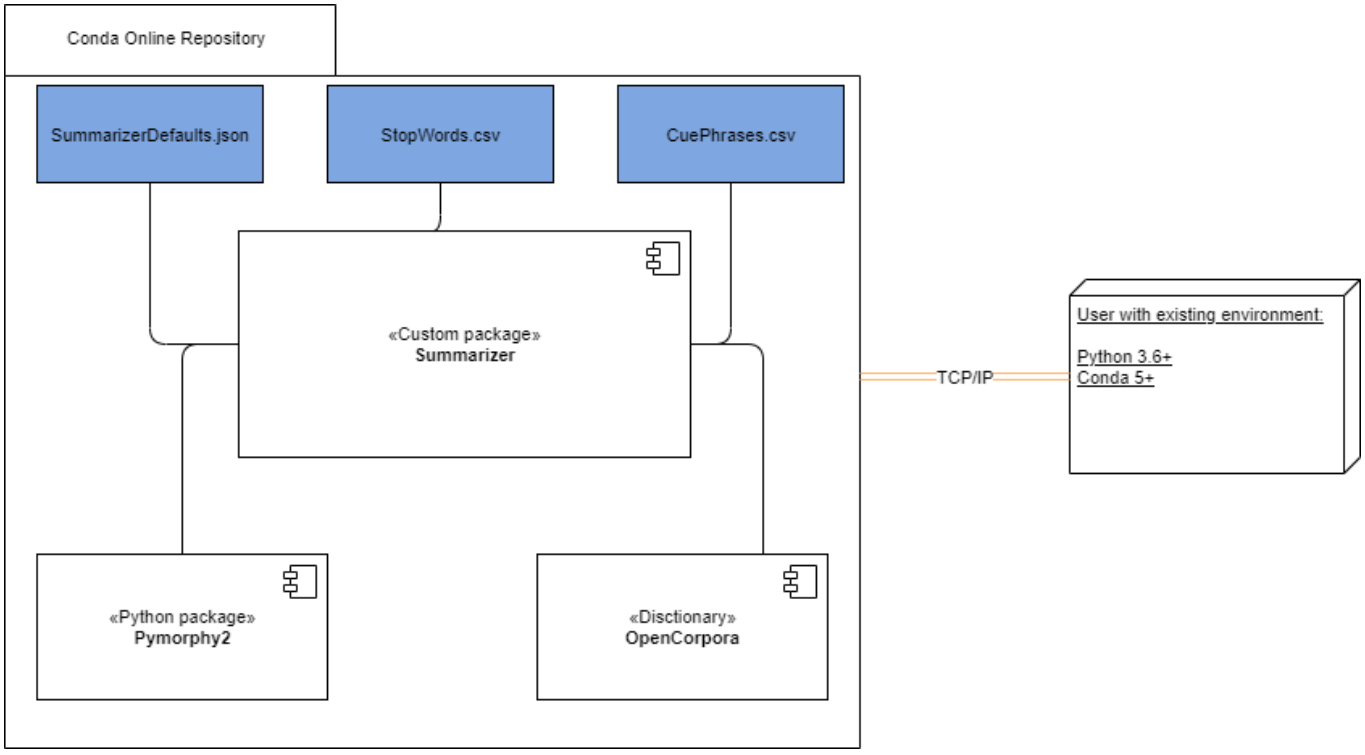


Рисунок 4.2 – UML-діаграма розгортання

Коментар до діаграми розгортання. Рішення для програмної бібіліотеки – це розгортання через існуючу систему пакетів Conda. Користувачу достатньо мануально додати посилання на ресурс та за допомогою простої команди інсталювати пакет. Після інсталювання пакет готовий до використання, хоча можливо користувач захоче відредагувати значення за замовчуванням чи додати якість нові стоп-слова.

Останньою важливою діаграмою, яка надасть трішки ясності щодо діаграми класів є діаграма послідовностей. На цій діаграмі представлена послідовність генерації реферату з використанням базового класу самаризатора та з використанням методу TextRank

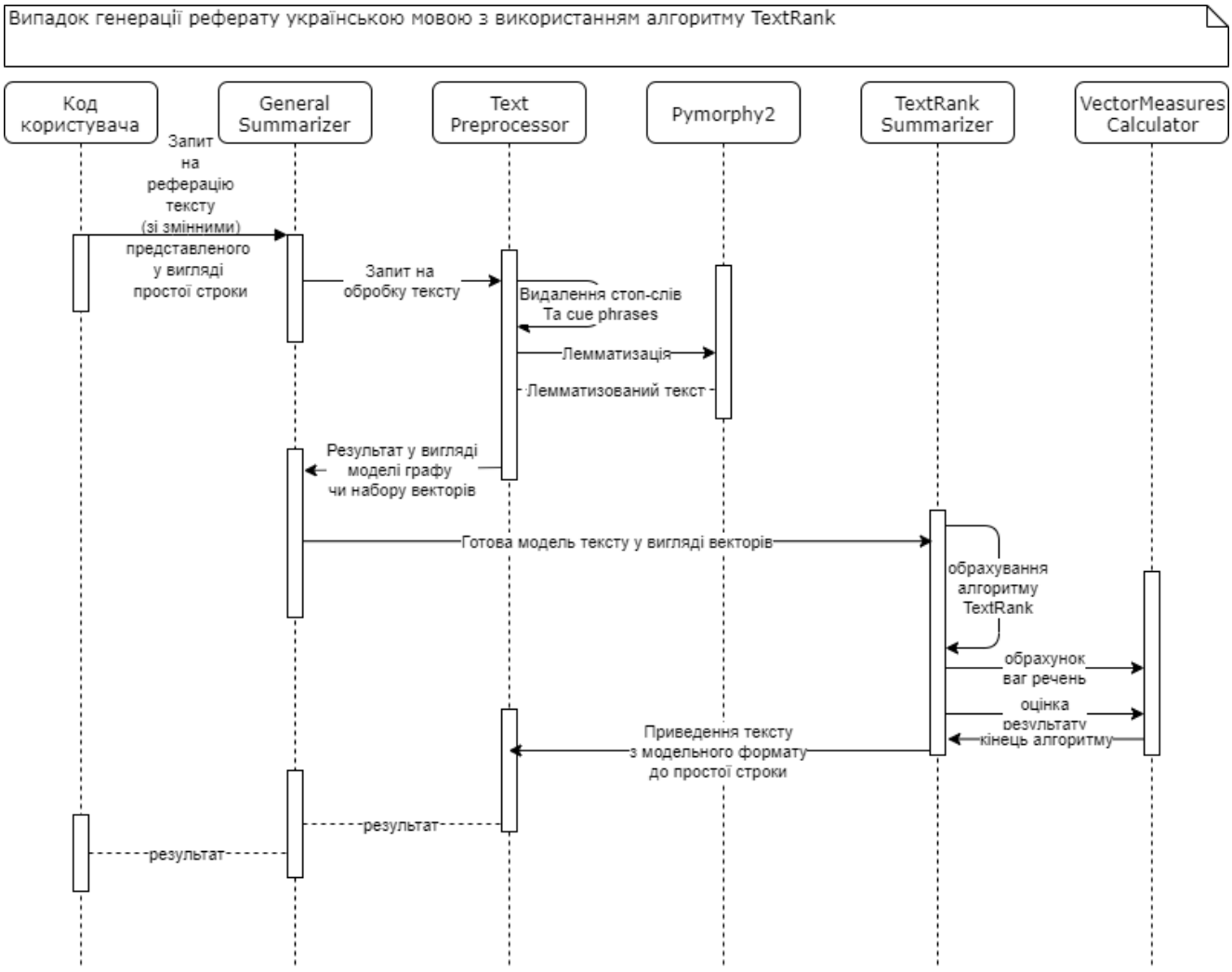


Рисунок 4.3 – UML-діаграма послідовностей

На цій діаграмі детально показана взаємодія між собою різних класів у пакеті. Як ми бачимо з неї, код користувача напряму спілкується виключно з класом, що імплементує інтерфейс Summarizer, що виступає фасадом більш складної системи та, навіть, використовує методи стороннього пакету у процесі (для лематизації). Також слід зазначити, що клас GeneralSummarizer повністю абстрагований від моделі представлення тексту і залежить виключно від класу TextPreProcessor. Тому тут виконана вимога щодо можливостей розширення системи і запропонована архітектура є досить непоганою з точки зору подальшої розробки.

4.3 КЕРІВНИЦТВО КОРИСТУВАЧА

Далі надамо керівництво для користувачів цього програмного пакету. Зазначимо, що пакет не має графічного інтерфейсу чи командного інтерфейсу та призначений для використання у програмному коді. Тому в керівництві зазначені тільки пункти пов’язані з встановленням пакету та використанням його методів.

4.3.1 ВСТАНОВЛЕННЯ ЗАСТОСУНКУ

Для коректної роботи застосунку, необхідно мати таке середовище:

- Conda[38] версій 5+;
- python 3.6[37];

Встановлення пакету протестовано на ОС Microsoft Windows 10. Для того, щоб встановити пакет, треба склонувати репозиторій, що розміщений за посиланням: https://github.com/arshakian/Summarizer.git. З командної лінії це можна зробити за допомогою команди

```
git clone  https://github.com/arshakian/Summarizer.git
```

Після успішного виконання даної команди створіть файл всередині створеної папки Summarizer з назвою environment.yml. його контент:

```
name: summarizer

channels:

- !!python/unicode

  'defaults'

dependencies:

- python

- scipy

- pymorphy2
```

Після цього достатньо виконати команду: conda env create – що створить середовище, в якому буде встановлений пакет та за допомогою простої залежності у python імпортуйте необхідні класи з пакету за допомогою оператора from, наприклад:

```
from summarizer import GeneralSummarizer
```

4.3.2 ВИКОРИСТАННЯ ЗАСТОСУНКУ

Пакет Summarizer містить у собі три публічних класи, які можна використовувати для отримання реферату з початкового тексту. Перерахуємо ці класи та їх методи, які можна викликати:

- GeneralSummarizer. Цей клас має такі методи:
    - summarize(text) – викликаючи метод без параметрів, будуть використані стандартні змінні – 0.85 як коефіцієнт загасання для TextRank, 0.001 як величина похибки вагових коефіцієнтів, TextRank та

LSA як метод (алгоритм описаний у 3.4) та 2 речення як розмір готового реферату;

- summarize (text, length, kf, epsilon, method) – можна перезаписати кожен з параметрів. Length – бажана довжина реферату у реченнях, KF – значення з крапкою коефіцієнта загасання, epsilon – розмір похибки, та method – що може приймати три значення – LSA, TextRank та null – в залежності від цього, програма буде використовувати той чи інший алгоритм роботи.

- LSASummarizer. Цей клас має такі методи:

- summarize(text) – викликаючи метод без параметрів, буде згенерований реферат довжиною у два речення методом латентно-семантичного аналізу;

- summarize (text, length) – якщо вказати напряму значення довжини, можна маніпулювати довжиною результуючого реферату.

- TextRankSummarizer. Цей клас має такі методи:

- summarize(text) – викликаючи метод без параметрів, будуть використані стандартні змінні – 0.85 як коефіцієнт загасання для TextRank, 0.001 як величина похибки вагових коефіцієнтів і 2 речення як розмір готового реферату. Реферат буде розраховуватись на базі алгоритму TextRank;

- summarize (text, length, kf, epsilon) – за допомогою цього методу можна змінювати коефіцієнт загасання чи величину похибки, як і бажану довжину реферату.

ВИСНОВКИ ДО РОЗДІЛУ 4

У цьому розділі розглядалась програмна складова розроблюваного рішення, зокрема, сформовано архітектурні вимоги, побудована архітектура застосунку через побудову деяких діаграм, таких як діаграма послідовностей,

діаграма розгортання та діаграма класів. Зважаючи на характер розроблюваного застосунку (а саме – бібліотека для подальшого використання, без інтерфейсу користувача) була надано керівництво для користувачів, в якому розглянуто особливості використання та встановлення застосунку.

Наступним кроком, зважаючи на вищевикладене та на запропонований алгоритм буде перевірка якості розробленого застосунку

5 ОЦІНКА ЯКОСТІ РОБОТИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 ТЕОРИТИЧНІ ВІДОМОСТІ ПРО МЕТОДИ ОЦІНЮВАННЯ ЯКОСТІ АВТОМАТИЧНОГО РЕФЕРАТУ

Для оцінки якості роботи алгоритму автоматичного реферування необхідний набір документів з доданими до них рефератами, які називають зразковими. Реферати, отримані за допомогою алгоритмів далі для зручності будемо називати автоматичними. В роботі[12] був запропонований пакет для оцінки якості алгоритмів автоматичного реферування текстів ROUGE, метрики якого мають високу кореляцію з людськими оцінками. Розглянемо докладніше основні метрики даного пакету. Зазначимо, що для оцінки пакетом ROUGE необхідно генерувати так звані зразкові реферати, тому виключне використання цього пакету не є доцільним. У розділі 2.2. були надані деякі статистичні показники, які побудовані використовуючи частотні показники – наприклад, косинус подібності, TF-IDF та інші. Незважаючи на певну обмеженість роботи цих показників (не розглядають семантичні особливості тексту), їх значення також можна розглядати щоб відкинути як суб’єктивність зразкових рефератів так і для оцінки взагалі без використання зразкового реферату.

5.1.1 ROUGE – N

Метрика ROUGE–N заснована на обчисленні кількості n-грам, що зустрічаються і в зразковому, і в автоматичному рефератах[12], а саме:

$$\mathrm{R}OUGE - N = \frac{Count_{match}(gram_{n)}}{Count_{ref}(gram_{n)}}, \qquad (5.1)$$

де $Count_{match}(gram_n)$ — кількість n-грам, що з'являються і в автоматичному рефераті, і в зразковому; $Count_{ref}\ (gram_n)$ — кількість n-грам в зразковому рефераті. У даній роботі використані метрика ROUGE - 2, що

використовує так звані біграми для підрахунку. Можна використовувати і інші метрики, зокрема уніграми, чи триграми.

5.1.2 ROUGE – L

Нехай X - послідовність слів зразкового реферату довжини n, Y - послідовність слів автоматичного реферату довжини m, а $LCS\ (X, Y)$ це довжина найбільшої загальної підпослідовності між X і Y, тоді:

$$P_{ics} = \frac{LCS(X,Y)}{m}$$

$$R_{ics} = \frac{LCS(X,Y)}{n}$$

$$ROUGE - L = \frac{2P_{ics}R_{ics}}{P_{ics} + R_{ics}}, \qquad (5.2)$$

Зазначимо, що в результаті виконання експерименту величина ROUGE–L майже без змін може використовувати оригінальний текст замість зразкового реферату.

5.1.3 ROUGE – S

Як і для метрики $ROUGE - L$ та і для цієї метрики характерні схожі результати не тільки для зразкових рефератів а і взагалі Нехай $SKIP2\ (X, Y)$ - кількість біграм з пропусками (термін – skip-bigram), що зустрічаються і в зразковому рефераті X довжини n слів, і в автоматичному рефераті Y довжини m, тоді:

$$P_{skip2} = \frac{SKIP2(X,Y)}{C_m^2}$$

$$R_{skip2} = \frac{SKIP2(X,Y)}{C_n^2}$$

$$ROUGE - S = \frac{2P_{ics}R_{ics}}{P_{ics} + R_{ics}}, \qquad (5.2)$$

5.1.4 Косинус подібності

Одним з основних параметрів автоматичної оцінки (тобто без зразкового реферату) є косинус подібності – у [13] цей параметр було запропоновано використовувати як основний для оцінки рефератів без суб'єктивності зразкового реферату. Косинус подібності розраховується по формулі (2.3). Можна використовувати і інші міри для порівняння, але для проведення експерименту та для оцінки його результатів достатньо буде приведених величин.

5.2 ОПИС ЕКСПЕРИМЕНТУ

Для проведення оцінки якості роботи застосунку було використано дані з інтернет-ЗМІ gazeta.ua[28], а саме новини з RSS-стрічки. Новостні статті гарно підходять для задачі реферування, а RSS-стрічка даного інтернет-ресурсу має як українську так і російську мову.

Датасет сфомовано з цих даних, використано 100 новин російською та 100 новин українською. Середній розмір тексту скаладає близько 20 речень, варіативність розміру тексту від 5 до 50 речень. Розмір реферату залежить від початкового розміру тексту, та визначається автоматично – для текстів довжиною більше 20 речень – 20%, для текстів до 20 речень розмір фіксований у 4 речення (до п'яти – 2)

Приклад новини зі стрічки:

«У шорт-лист кінопремії BAFTA потрапив ігровий фільм "Анна" Декеля Беренсона.Стрічка змагатиметься у номінації "Найкращий британський короткометражний фільм", повідомили на сайті Британської кіноакадемії. Нагороду в цій категорії отримали 10 кінострічок. ЧИТАЙТЕ ТАКОЖ: "Наші котики": з'явився трейлер неполіткоректної комедії про Донбас Фільм "Анна" знімали у копродукції України, Ізраїлю та Британії. Його представили 24 травня 2019 року на 72-му Каннському міжнародному кінофестивалі. Стрічка увійшла

до програми Національного конкурсу 48-го Київського міжнародного кінофестивалю "Молодість".В центрі сюжету фільму історія про матір-одиначку на ім'я Анна. Вона живе на охопленому війною Донбасі й виховує 16-річну дочку. Головна героїня працює на м'ясопереробному комбінаті і мріє про краще життя.Український режисер Сергій Лозниця отримав державну премію імені Олександра Довженка за повнометражну стрічку "Донбас". Міністр культури, молоді та спорту Володимир Бородянський вручив чоловіку нагороду.»

Приклад отриманого реферату реферату:

«У шорт-лист кінопремії BAFTA потрапив ігровий фільм "Анна" Декеля Беренсона. В центрі сюжету фільму історія про матір-одиначку на ім'я Анна.»

Зазначимо, що при формуванні датасету з тексту були видалені HTML-теги. Однак, деякі елементи тексту, як наприклад речення, що починаються з «ЧИТАЙТЕ ТАКОЖ:» навмисно не видалені для перевірки роботи, так як ці вставки зазвичай не пов’язані семантично з текстом.

5.3 ОЦІНЮВАНКА РЕЗУЛЬТАТІВ ЕКСПЕРИМЕНТУ

Результати обрахунків для реферату величин, які описані у 5.1 з використанням датасету з розділу 5.2

Таблиця 5.1 – Середьні значення оцінок алгоритмів роботи застосунку

Алгоритм	ROUGE-2	ROUGE – L	ROUGE-S	Cos θ
TextRank	0.08	0.12	0.04	0.54
LSA	0.08	0.11	0.03	0.40
TextRank + Pymorphy2	0.10	0.13	0.05	0.59

Продовження таблиці 5.1

LSA + Pymorhpy2	0.08	0.12	0.04	0.46
Hybrid + Pymorhpy2	0.10	0.13	0.05	0.6

Наведемо також таблицю з максимальними значеннями для розроблюваних алгоритмів

Таблиця 5.2 – Максимальні значення для оцінки алгоритмів роботи застосунку

Алгоритм	ROUGE-2	ROUGE – L	ROUGE-S	Cos θ
TextRank	0.41	0.36	0.32	0.84
LSA	0.33	0.33	0.28	0.71
TextRank + Pymorphy2	0.43	0.37	0.33	0.85
LSA + Pymorhpy2	0.35	0.39	0.26	0.73
Hybrid + Pymorhpy2	0.43	0.39	0.33	0.85

Як бачимо, результати експерименту досить сильно відрізняються від обраного алгоритму. Також, варто відмітити результати отримані з використанням бібліотеки Pymorhpy2[20] – в середньому вони надають кращі результати, хоча як бачимо інколи існують ситуації коли використання надлишкової підготовки погіршує значення однієї з метрик. У роботі[15] зазначено середнє значення ROUGE-2 метрики для алгоритму TextRank для англійської мови у 0.42, але варто зазначити, що використовувася інший набір даних, зразкових рефератів, та розміри реферату зазвичай допускались до 20%

від початкового тексту. Тому результати експерименту можна вважати вдалими.

ВИСНОВКИ ДО РОЗДІЛУ 5

В даному розділі були наведені загальні відомості про оцінювання роботи алгоритмів екстративного реферування, як з використанням зразкових рефератів, так і без них. Були представлені відомості про пакет оцінок ROUGE (а саме ROUGE-N, ROUGE-L та ROUGE-S), згадано про інші метрики для оцінювання екстративного реферату.

В результаті проведення експерименту було показано, що використання лематизації та підготовки тексту за допомогою бібліотеки pymorphy2 надає кращі як середні, так і максимальні показники для обраного набору даних. Подальші покращення роботи алгоритму можливі саме у цьому напрямку. Найкращим з протестованих алгоритмів можна вважати гібридній підхід описаний у 3.4, але він є найбільш ресурсоємким.

6 РОЗРОБКА СТАРТАП-ПРОЕКТУ

6.1 ОПИС ІДЕЇ СТАРТАП-ПРОЕКТУ

Зважаючи на те, що розроблюваний у розділі 4 програмний продукт виступає як вільнопоширювана бібліотека для вирішення задачі реферування, ідея стартап-проекту буде використовувати розроблене програмне забезпечення, але при цьому буде окремим програмним продуктом – сервісом для автоматичного реферування текстів для інших програмних продуктів з системою монетизації за кількість запитів до сервісу. Зазначимо, що розроблюваний сервіс бажано поставляти по схемі Software-as-a-service, так як саме вона ідеально підходить для реалізації маленьких підзадач, як, наприклад, реферування текстів. Наведемо дані про ідею у вигляді таблиці:

Таблиця 6.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Розробити застосунок який буде працювати по схемі SaaS та буде надавати по запиту автоматичні реферати по заданому тексту чи для певного потоку даних – динамічно розбиваючи його на підтексти	1. Реферування потоку даних написаних українською чи російською мовою	Не потрібно імплементувати код власноруч, достатньо виклика стороннього сервісу який імплементує цю задачу. Автоматичне розбиття потоку на окремі тексти.
	2. Реферування по запиту для конкретного тексту	Не потрібно імплементувати код власноруч, достатньо виклика стороннього сервісу який імплементує цю задачу

6.2 АНАЛІЗ РИНКОВИХ МОЖЛИВОСТЕЙ СТАРТАП-ПРОЕКТУ

Розробка стартапу можлива після проведення аналізів ринкових можливостей, формування потенційних груп користувачів та аналізу існуючих пропозицій. Важливо також розглянути потенційних конкурентів на цьому ринку. Результати проведених аналізів можна скласти у вигляді табличок, як це і буде представлено далі.

Таблиця 6.2 – Попередня характеристика ринку стартап-проекту

№ п/п	*Показники стану ринку (найменування)*	*Характеристика*
1	Кількість головних гравців, од	0
2	Загальний обсяг продаж, грн/ум.од (через витрати на продукти-замінники)	2 500 000
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Юридичні обмеження (наприклад, GDPR)
5	Специфічні вимоги до стандартизації та сертифікації	Вимоги до програмного забезпечення. Юридичні
6	Середня норма рентабельності в галузі (або по ринку), %	50 %

Банківський відсоток на вкладення – 20% (у гривні), середня норма рентабельності – 50%, тому є сенс вкладати кошти в цей проект. Зважаючи на кількість граців та відсутність прямих конкурентів цей ринок є досить привабливим для потенційних інвесторів. Звісно, деякі юридичні обмеження для певних територій можуть стати проблемою, але в цілому зважаючи на рентабільність не стане великою проблемою.

Розглянемо далі потенційних клієнтів нашого стартапу. Для цього побудуємо харатеристику потенційних клієнтів у вигляді таблички.

Таблиця 6.3 – Характеристика потенційних клієнтів застосунку

№ п/п	*Потреба, що формує ринок*	*Цільова аудиторія (цільові сегменти ринку)*	*Відмінності у поведінці різних потенційних цільових груп клієнтів*	*Вимоги споживачів до товару*
1	Великі об’єми потоків даних для опрацювання	Компанії, що оброблюють потоки текстових даних, наприклад FinTech компанії, що оброблюють потоки новин	відсутні	Бистродійне та надійне рішення для зменьшення об’єму даних у потоці без потенційного зменьшення якості даних
2	Великі об’єми текстів даних	Компанії, які щоденно формують реферати з початкових даних, зокрема ЗМІ	відсутні	Надійне та якісне рішення для скорочення текстів по запиту

Зважаючи на те, що клієнти – це компанії середьного та великого розміру, є сенс розроблювати додаток по моделі B2B – проводити маркетингову політику зваючи на те, що користувачами будуть тільки компанії, а не індивідуальні користувачі. Залишилось розглянути моделі конкурентів, і це

буде зроблено через порівняльну характеристику надану за методом п’яти сил М. Портера

Таблиця 6.4 – Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	*Потенційні конкуренти*	*Постачальники*	*Клієнти*	*Товари-замінники*
Складові аналізу	Відсутні	Існуючі рішення, що потребують додаткової імплементації юридичні питання	PaaS-постачаль-ники Хостинги	Кількість користувачів та їх зацікавленість у платній версії	Часткова заміна, вища якість
Висновки	Не інтенсивна	- можливо є потенційні конкуренти у інших країнах - можливі потенційні конкуренти у суміжних галузях	Так, погодження умов співпраці	Так, Якість застосунку, зацікавленість в користуванні	Так, велика вігорідність що деякі корпортативні клієнти будуть використовувати Продукти-замінники

Висновок: принципово можливо працювати на ринку з огляду на конкурентну ситуацію. Щоб бути конкурентоспроможним на ринку, проект повинен мати такі характеристики (сильні сторони):

- нижчу ціну аніж за імплементацію власного рішення для великого бізнесу

- висока якість застосунку

- таргетованість та клієнтоорієнтовану рекламу компанію

Фінальним етапом ринкового аналізу є складання SWOT-аналізу, матриці сильних та слабких сторін продукту, загроз для стартапу та потенційних можливостей. Перелік ринкових загроз та ринкових можливостей складається на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та ринкові можливості є наслідками (прогнозованими результатами) впливу факторів, і, на відміну від них, ще не є реалізованими на ринку та мають певну ймовірність здійснення. Наприклад: зниження доходів потенційних споживачів – фактор загрози, на основі якого можна зробити прогноз щодо посилення значущості цінового фактору при виборі товару та відповідно, – цінової конкуренції (а це вже – ринкова загроза).

Таблиця 6.5 – SWOT-аналіз стартап-проекту

Сильні сторони: - Збільшення об’ємів текстових даних та актуалізація задачі автоматичного реферування - Відсутність прямих конкурентів (а саме компаній які пропонують вирішення цієї задачі по схемі SaaS)	Слабкі сторони: - B2B-модель потребує багато витрат на маркетинг та на відділ продажів - Великі затрати на розробку - Високі ризики неокупності

Можливості:	Загрози:
- Удешевіння операційних витрат через зменьшення ціни на хостинги, платформи, сервери - Великий кількість клієнтів з різних сфер та локацій - Зростання ринку сервісів SaaS	- Обмеженість зацікавлених користувачів - Не готовність користувачів платити за проект - Ціноутворення, при якому вигідніше буде мати власне вбудоване рішення

Зважаючи на аналіз, доцільно буде перейти до розробки маркетингової концепції та стратегії.

6.3 РОЗРОБЛЕННЯ МАРКЕТИНГОВОЇ СТРАТЕГІЇ СТАРТАПУ

Першим кроком у розробці маркетингової концепції є формування маркетингової концепції товару. Наведо відомості про цю концепцію у вигляді таблиці:

Таблиця 6.6 – Визначення ключових переваг продукту

№ п/п	*Потреба*	*Вигода, яку пропонує товар*	*Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити*
1	Автоматичне реферування текстів та потоків	Найбільш ефективні методи екстрактивного реферування для текстів російською та українською мовами	Найдосконаліші методи для екстрактивного реферування в розробленій бібліотеці.
2	Економія часу та	Не потрібно витрачати час	Система доступна після швидкої процедури реєстрації, не потрібно

	видатків на розробку	на власну розробку методів для обробки та реферування текстів	розробляти з нуля, підтримувати сервери та інше

Далі для визначення маркетингової стратегії треба узгодити два пункти – ціноутворення та канали для продажів. Зазначимо, що ціноутворення зазвичай визначають тільки на етапі фінансово-економічного аналізу проекту, та на даному етапі можна лише визначити загальну ціну, використовуючи експертну думку. Ціноутворення даного продукту має бути вигідним для користувача, тому ми розглядаємо ціну для одного користувача за запит таку, що для умовної кількості запитів на місяць, добуток ціни за ліцензію на кількість запитів був меньшим за ціну розробки власного застосунку разом з операційними коштами.

Щодо ж каналів продажу, то зважаючи на модель B2B та орієнтацію на корпоративних клієнтів, основним каналом продажів будуть зустрічі та презентації кожному з потенційних клієнтів, з використанням спеціального програмного забезпечення для управління потенційними клієнтами. Для маркетологів та відділу продажів, можна надати ще концепцію макретингових комунікацій. В цій концепції основним буде орієнтація на модерновий підхід SaaS для роботи застосунку, основне завдання – переконати потенційного клієнта у рентабельності використання цього рішення порівнючи з с сервісами-замінниками.

ВИСНОВКИ ДО РОЗДІЛУ 6

В даному розділі було проаналізовано стартап-продукт, який потенційно можна побудувати на базі програмного рішення з розділу 4. З наведених у

розділі аналітичних таблиць, а саме зі SWOT-аналізу, характеристики потенційний користувачів, аналізу конкуренції в галузі за системою п’яти сил М. Портера, характеристики ринку можна сказати, що наведений стартап-продукт є досить нішевим, високомаржинальним та унікальним продуктом без прямих конкурентів, лише з сервісами-замінниками. Для початкової версії є сенс розробити лише мінімальний продукт та залучені кошти виділяти на макретинг та на відділ продажу. Зважаючи на ринок та на характеристики продукту, розроблювальний стартап-продукт досить швидко вийде на прибутковість.

ВИСНОВКИ

В даній роботі ми розглянули існуючі математичні методи та програмні методи вирішення задачі реферування текстів українською та російською мовами. Зважаючи на аналіз та вибір методів і постановку задачі, було побудовано програмне забезпечення, що реалізує одні з найпопулярніших алгоритмів екстрактивної реферації TextRank та LSA, які використовують різні математичні моделі тексту; використовує морфологічний аналізатор pymorphy2 для значного покращення результатів на етапі попередньої обробки тексту; реалізовує запропонований алгоритм, який надає результат, який отримано через LSA чи через TextRank в залежності від косинусу подібності; надає потенційно кращі результати за методи без морфологічного аналізатору.

В математичному розділі були розглянуті основні математичні методи представлення тексут, а саме 1-hot word enconding, Bag-Of-Words, Bag-Of-Terms, векторне представлення тексту та представлення тексту у вигляді графу. Зважаючи на представлені моделі тексту, для деяких з них було розглянуто приклади вагових функцій.

В результаті проведення експерименту було показано, що використання лематизації та підготовки тексту за допомогою бібліотеки pymorphy2 надає кращі як середні, так і максимальні показники для обраного набору даних.

Наукова новизна полягає у розробці власного алгоритму реферування текстів та його імплементація для вирішення задачі реферування текстів, написаних українською та російською мовами.

СПИСОК ВИКОРИСТАННИХ ДЖЕРЕЛ

1) If you could print out the whole Internet, how many pages would it be? [Electronic Resource] – Access Mode: World Wide Web: https://www.washingtonpost.com/news/the-intersect/wp/2015/05/18/if-you-could-print-out-the-whole-internet-how-many-pages-would-it-be/. – Title from the screen.

2) Агеев М. С., Добров Б. В., Лукашевич Н. В. Автоматическая рубрикация текстов: методы и проблемы // Учёные записки Казанского государственного университета. Серия Физико-математические науки. — 2008. — Т. 150, № 4. — С. 25–40.

3) Воронцов К. В., Потапенко А. А. Регуляризация вероятностных тематических моделей для повышения интерпретируемости и определения числа тем // Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной Международной конференции «Диалог» (Бекасово, 4–8 июня 2014 г.). — Вып. 13 (20). — М: Изд-во РГГУ, 2014. — С. 676–687.

4) Bolelli L., Ertekin S., Giles C. L. Topic and trend detection in text collections using latent Dirichlet allocation // ECIR. — Vol. 5478 of Lecture Notes in Computer Science. — Springer, 2009. — pp. 776–780

5) Chien J.-T., Chang Y.-L. Bayesian sparse topic model // Journal of Signal Processessing Systems. — 2013. — Vol. 74. — pp. 375–389.

6) Dempster A. P., Laird N. M., Rubin D. B. Maximum likelihood from incomplete data via the EM algorithm // J. of the Royal Statistical Society, Series B. — 1977. — no. 34. — pp. 1–38.

7) Blei D. M., Jordan M. I. Modeling annotated data // Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval. — New York, NY, USA: ACM, 2003. — Pp. 127–134.

8) Bodrunova S., Koltsov S., Koltsova O., Nikolenko S. I., Shimorina A. Interval semisupervised LDA: Classifying needles in a haystack // MICAI (1) / Ed.

by F. C. Espinoza, A. F. Gelbukh, M. Gonzalez-Mendoza. — Vol. 8265 of Lecture Notes in Computer Science. — Springer, 2013. — pp. 265–274

9) Автоматическая обработка текстов на естественном языке и анализ данных : учеб. пособие / Большакова Е.И., Воронцов К.В., Ефремова Н.Э., Клышинский Э.С., Лукашевич Н.В., Сапин А.С. — М.: Изд-во НИУ ВШЭ, 2017. — 269 с. ISBN 978–5–9909752-1-7

10) New Methods in Automatic Extracting, Journal of the ACM / Edmundson, H.P., 1969 – pp 264-285.

11) ІІ Всеукраїнська науково-практична конференція молодих вчених та студентів «Інформаційні системи та технології управління – ІСТУ-2019». Секція кафедри автоматизованих систем обробки інформації і управління. Матеріали конференції. – Київ. – 2019. 16 – 17 травня 2019р. – 130 с.

12) C.-Y. Lin. ROUGE: A package for automatic evaluation of summaries// Proceedings of ACL Text Summarization Branches Out Workshop. – 2004. – С. 74–81,

13) Automatic Summary Evaluation without Human Models / Annie Louis [Electronic Resource] – Mode of access: World Wide Web: https://pdfs.semanticscholar.org/98a5/2c172a49b32a710a116436b025d441bc0ee5.pdf – Title from the screen

14) Miso-Belica / Sumy [Electronic Resource] – Mode of access: World Wide Web: https://github.com/miso-belica/sumy – Title from the screen

15) R. Mihalcea and P. Tarau. TextRank: Bringing Order into Texts// Proc. of the 9th Conf. on Empirical Methods in Natural Language Processing. – 2004. – С. 404–411

16) J. Steinberger and K. Jezek. Using Latent Semantic Analysis in Text Summarization and Summary Evaluation// Proc. of ISIM. – 2004. – С. 93–100.

17) Automatic Summarization of News Articles using TextRank [Електронний р есурс] / A. Zadbuke, S. Pimenta, D. Padwal, V. Wangikar // International Journal of Advanced Research in Computer Science and Software

Engineering. 2016. №6. — S. 124-127. — Режим доступу : http://www.ijarcsse.com/docs/papers/Special_Issue/iconect2016/sfcs02.pdf.

18) Automatic Text Document Summarization using Semantic-based Analysis / Chandra Shekhar Yadav — Режим доступу: https://arxiv.org/abs/1811.06567

19) Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krys Kochut. 2017. Text Summarization Techniques: A Brief Survey. In Proceedings of arXiv, USA, July 2017, 9 pp.

20) Korobov M.: Morphological Analyzer and Generator for Russian and Ukrainian Languages // Analysis of Images, Social Networks and Texts, pp 320-332 (2015).

21) Harris, David and Harris, Sarah (2012-08-07). Digital design and computer architecture (2nd ed.). San Francisco, Calif.: Morgan Kaufmann. p. 129. ISBN 978-0-12-394424-5.

22) Speech and Language Processing An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition by Daniel Jurafsky, James H. Martin

23) Mani, Kaustubh & Verma, Ishan & Meisheri, Hardik & Dey, Lipika. (2018). Multi-Document Summarization Using Distributed Bag-of-Words Model. 672-675. 10.1109/WI.2018.00-14.

24) G. Salton. Automatic Text Processing. Addison-Wesley Publishing Company, Inc., Reading, MA, 1989.

25) Thi Ngoc Quynh Do - A graph model for text analysis and text mining, 2012

26) P. B. Baxendale - Machine-made index for technical literature: an experiment, 1958

27) Аршакян Г.Д. «Огляд підходів та методів автоматичного реферування тексту» / Ю.О. Олійник // Матеріали III всеукраїнської науково-практичної конференції молодих вчених та студентів «Інформаційні системи та технології

управління» (ІСТУ-2019) – м. Київ.: НТУУ «КПІ ім. Ігоря Сікорського», 20-22 листопада 2019 р.

28) Основні новини дня України та світу [Electronic Resource] – Mode of access: World Wide Web: https://gazeta.ua – Title from the screen

29) Каніщева О.В. Використання карт відношень (TRM) для автоматичного реферування / О.В. Каніщева // Вісник Національного університету "Львівська політехніка". – 2013. – № 770 : Інформаційні системи та мережі. – С. 108 – 122.

30) Bill Camarda – Using Microsoft Word / Bill Camarda. – 2002. – pp. 326–330

31) SMMRY - Summarize articles, text, websites, essays and documents [Electronic Resource] – Mode of access: World Wide Web: https://smmry.com – Title from the screen

32) Text Compactor: Free Online Automatic Text Summarization Tool [Electronic Resource] – Mode of access: World Wide Web: https://textcompactor.com – Title from the screen

33) Главред β [Electronic Resource] – Mode of access: World Wide Web: https://glvrd.ru – Title from the screen

34) sumy · PyPI [Electronic Resource] – Mode of access: World Wide Web: https://pypi.org/project/sumy/ – Title from the screen

35) letiantian/TextRank4ZH [Electronic Resource] – Mode of access: World Wide Web: https://github.com/letiantian/TextRank4ZH – Title from the screen

36) OpenCorpora: открытый корпус русского языка [Electronic Resource] – Mode of access: World Wide Web: https://github.com/letiantian/TextRank4ZH – Title from the screen

37) Welcome to Python.org [Electronic Resource] – Mode of access: World Wide Web: https://www.python.org/ – Title from the screen

38) Conda — Conda documentation [Electronic Resource] – Mode of access: World Wide Web: https://conda.io/en/latest/ – Title from the screen

ДОДАТОК А – КОД ПРОГРАМНОГО ЗАСТОСУНКУ

GeneralSummarizer.py

```
from itertools import combinations

from pymorphy2 import MorphAnalyzer

from TextPreprocessor import sent_tokenizer_ru, word_tokenizer, stop_words_ru, sent_tokenizer_ua, stop_words_ua

import math

from TextGraph import rank_graph

from numpy import dot

from numpy.linalg import norm

from VectorMeasuresCalculator import tfidf, get_cosine, text_to_vector

import LSASummarizer


def similarity(s1, s2):
    n1 = norm(s1)
    n2 = norm(s2)
    if not n1 or not n2:
        return 0.0
    return dot(s1, s2)/(n1*n2)


def text_rank(text, language):
    sentences = []
```

```
a = []
if (language == 'ukrainian'):
    morph = MorphAnalyzer(lang='uk')
    sentences = sent_tokenizer_ua(text)
    if len(sentences) < 2:
        s = sentences[0]
        return [(1, 0, s)]
    a = tfidf(text, language, sent_tokenizer_ua, stop_words_ua)
else:
    morph = MorphAnalyzer()
    sentences = sent_tokenizer_ru(text)
    if len(sentences) < 2:
        s = sentences[0]
        return [(1, 0, s)]
    a = tfidf(text, language, sent_tokenizer_ru, stop_words_ru)

pairs = combinations(range(len(sentences)), 2)
scores = [(i, j, similarity(a[i, :], a[j, :])) for i, j in pairs]
scores = filter(lambda x: x[2], scores)

pr = rank_graph(scores)
```

```
    return sorted(((i, pr[i], s) for i, s in enumerate(sentences) if i in pr),
                  key=lambda x: pr[x[0]], reverse=True)  # Сортировка по убыванию ранга тройки



def summarize(text, language, n=5):
    tr = text_rank(text, language)
    top_n = sorted(tr[:n])
    text_rank_result = ' '.join(x[2] for x in top_n)
    lsa_result = LSASummarizer.summarize(text, language, n)
    cosine_text_rank = get_cosine(text_to_vector(text), text_to_vector(text_rank_result))
    cosine_lsa=get_cosine(text_to_vector(text), text_to_vector(lsa_result))
    if cosine_lsa > cosine_text_rank:
        return lsa_result
    return text_rank_result
```

LSASummarizer.py

```
from pymorphy2 import MorphAnalyzer
import numpy
from numpy.linalg import svd
from TextPreprocessor import sent_tokenizer_ru, word_tokenizer, stop_words_ru, sent_tokenizer_ua, stop_words_ua
import math
```

```
def create_dictionary(text, morph, stop_words):
    words = set(morph.parse(word)[0].normalized for word in word_tokenizer.tokenize(
        text.lower()) if word not in stop_words)
    return dict((w, i) for i, w in enumerate(words))


def create_matrix(text, sent_tokenizer, morph, dictionary):
    sentences = sent_tokenizer(text)

    words_count = len(dictionary)
    sentences_count = len(sentences)

    matrix = numpy.zeros((words_count, sentences_count))
    for col, sentence in enumerate(sentences):
        for word in word_tokenizer.tokenize(sentence.lower()):
            word = morph.parse(word)[0].normalized
            if word in dictionary:
                row = dictionary[word]
                matrix[row, col] += 1
    rows, cols = matrix.shape
```

```
    if rows and cols:
        word_count = numpy.sum(matrix)
        for row in range(rows):
            unique_word_count = numpy.sum(matrix[row, :])
            for col in range(cols):
                if matrix[row, col]:
                    matrix[row, col] = unique_word_count/word_count
    else:
        matrix = numpy.zeros((1, 1))
    return matrix



def compute_ranks(matrix, n):
    u_m, sigma, v_m = svd(matrix, full_matrices=False)
    powered_sigma = tuple(s**2 if i < n else 0.0 for i, s in enumerate(sigma))
    ranks = []
    for column_vector in v_m.T:
        rank = sum(s*v**2 for s, v in zip(powered_sigma, column_vector))
        ranks.append(math.sqrt(rank))
    return ranks
```

```
def summarize(text, language, n=5):
    morph = MorphAnalyzer()
    sent_tokenizer = sent_tokenizer_ru
    stop_words = stop_words_ru

    if language == 'ukrainian':
        morph = MorphAnalyzer(lang='uk')
        sent_tokenizer = sent_tokenizer_ua
        stop_words = stop_words_ua

    d = create_dictionary(text, morph, stop_words)
    m = create_matrix(text, sent_tokenizer, morph, d)
    r = compute_ranks(m, n)
    sentences = sent_tokenizer(text)
    rank_sort = sorted(((i, r[i], s) for i, s in enumerate(
        sentences)), key=lambda x: r[x[0]], reverse=True)
    top_n = sorted(rank_sort[:n])
    return ' '.join(x[2] for x in top_n)
```

RandomSummarizer.py

```
import random
from TextPreprocessor import sent_tokenizer_ru, sent_tokenizer_ua
```

```
def summarize(text, language, n):
    sent_tokenizer = sent_tokenizer_ru
    if language == 'ukrainian':
        sent_tokenizer = sent_tokenizer_ua
    sentences = sent_tokenizer(text)
    ratings = list(range(len(sentences)))
    random.shuffle(ratings)
    rand_sent = sorted((r, s) for r, s in zip(ratings, sentences))
    sent_n = rand_sent[:n]
    return ' '.join(s[1] for s in sent_n)
```

TextPreprocessor.py

```
# Инициализация всего необходимого для предобработки
from nltk.tokenize.punkt import PunktSentenceTokenizer, PunktParameters
from nltk.tokenize import RegexpTokenizer
import csv
import io


def get_stop_words(language):
    stopwords = []
    filename = 'stop_words_ru.csv'
    if language == 'ukrainian':
        filename = 'stop_words_ua.csv'
```

```
    with io.open(filename, 'r', encoding="utf-8") as file:
        for row in csv.reader(file):
            stopwords.append(row[0])
    return stopwords


def get_abbreviation(language):
    if language == 'ukrainian':
        return ['тис', 'грн', 'т.я', 'вул', 'сек', 'хв', 'обл', 'кв', 'пл', 'напр', 'гл', 'і.о', 'зам']
    return ['тыс', 'руб', 'т.е', 'ул', 'д', 'сек', 'мин', 'т.к', 'т.н', 'т.о', 'ср', 'обл', 'кв', 'пл',
            'напр', 'гл', 'и.о', 'им', 'зам', 'гл', 'т.ч']


punkt_param_ru = PunktParameters()
abbreviation_ru = get_abbreviation('russian')
punkt_param_ru.abbrev_types = set(abbreviation_ru)
sent_tokenizer_ru = PunktSentenceTokenizer(punkt_param_ru).tokenize
stop_words_ru = get_stop_words('russian')


punkt_param_ua = PunktParameters()
abbreviation_ua = get_abbreviation('ukrainian')
punkt_param_ua.abbrev_types = set(abbreviation_ua)
sent_tokenizer_ua = PunktSentenceTokenizer(punkt_param_ru).tokenize
stop_words_ua = get_stop_words('ukrainian')
```

```
word_tokenizer = RegexpTokenizer(r'\w+')
```

TextRankSummarizer.py

```
from itertools import combinations
from pymorphy2 import MorphAnalyzer
from TextPreprocessor import sent_tokenizer_ru, word_tokenizer, stop_words_ru, sent_tokenizer_ua, stop_words_ua
import math
from TextGraph import rank_graph


def similarity(s1, s2):
    if not len(s1) or not len(s2):
        return 0.0
    return len(s1.intersection(s2))/(math.log(len(s1)+1) + math.log(len(s2)+1))



def text_rank(text, language):
    sentences = []
    words = []

    if (language == 'ukrainian'):
        morph = MorphAnalyzer(lang='uk')
```

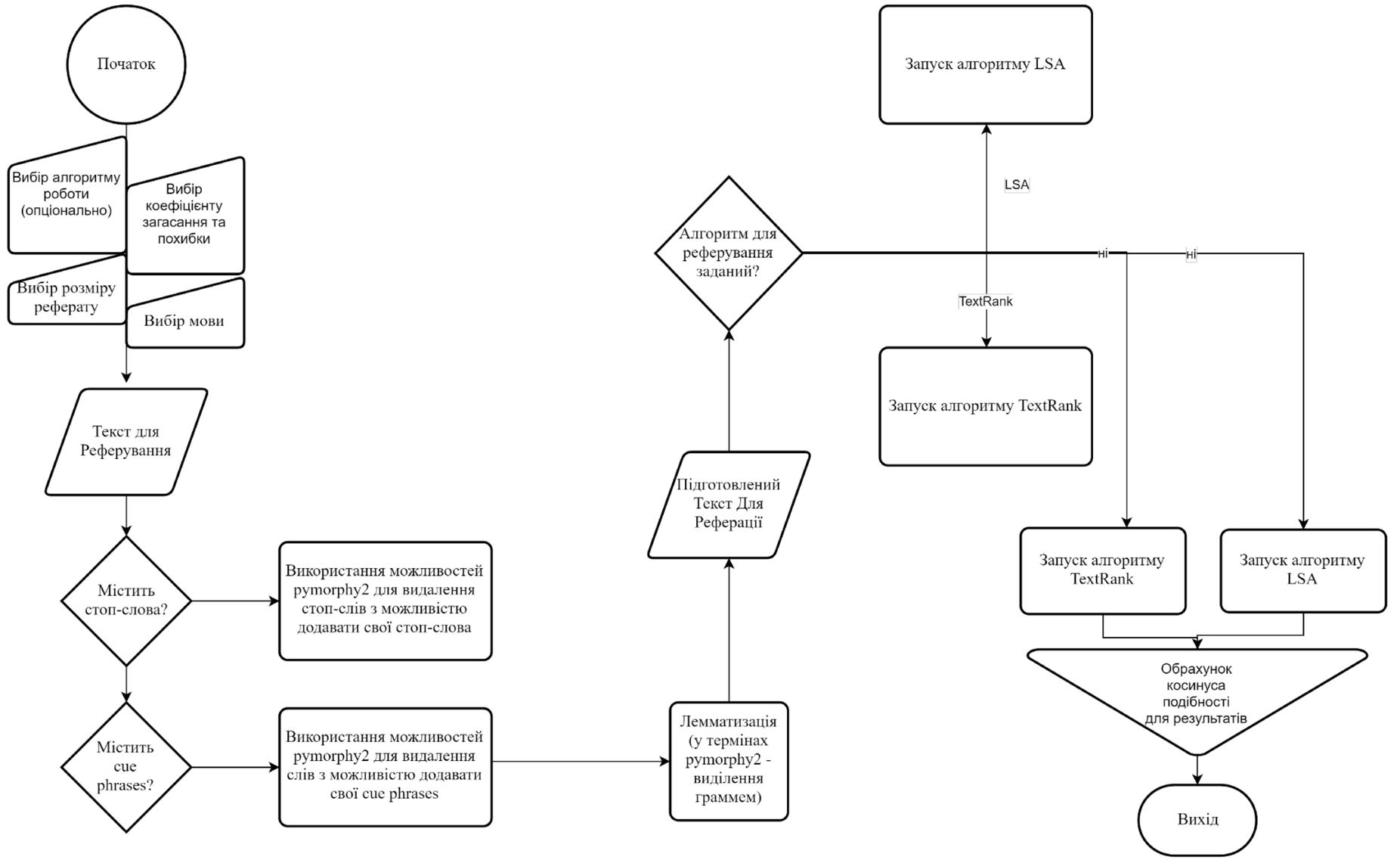
```
        sentences = sent_tokenizer_ua(text)
        if len(sentences) < 2:
            s = sentences[0]
            return [(1, 0, s)]
        words = [set(morph.parse(word)[0].normalized for word in
word_tokenizer.tokenize(sentence.lower())
                 if word not in stop_words_ua) for sentence in sentences]
    else:
        morph = MorphAnalyzer()
        sentences = sent_tokenizer_ru(text)
        if len(sentences) < 2:
            s = sentences[0]
            return [(1, 0, s)]
        words = [set(morph.parse(word)[0].normalized for word in
word_tokenizer.tokenize(sentence.lower())
                 if word not in stop_words_ru) for sentence in sentences]

    pairs = combinations(range(len(sentences)), 2)
    scores = [(i, j, similarity(words[i], words[j])) for i, j in pairs]
    scores = filter(lambda x: x[2], scores)
    pr = rank_graph(scores)
```

```
    return sorted(((i, pr[i], s) for i, s in enumerate(sentences) if i in pr),
                  key=lambda x: pr[x[0]], reverse=True)


def summarize(text, language, n=5):
    tr = text_rank(text, language)
    top_n = sorted(tr[:n])
    return ' '.join(x[2] for x in top_n)
```

ДОДАТОК Б - Схема роботи алгоритму



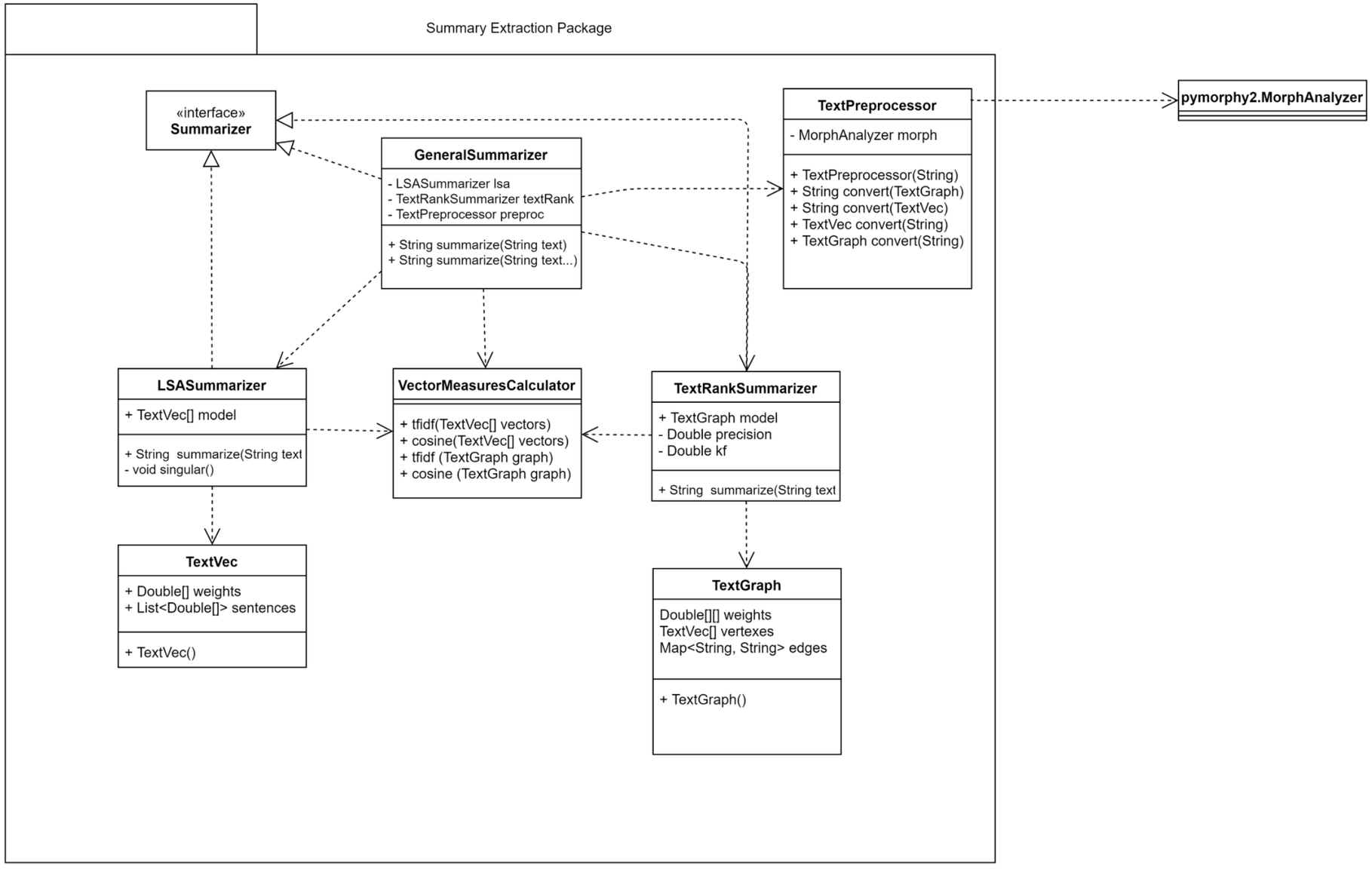
Демонстраційний плакат до магістерської дисертації

Схема роботи алгоритму

Виконав студент гр. ІП-з81мп Аршакян Г.Д.

Керівник Олійник Ю.О.

ДОДАТОК В - Діаграма класів розробленого застосунку



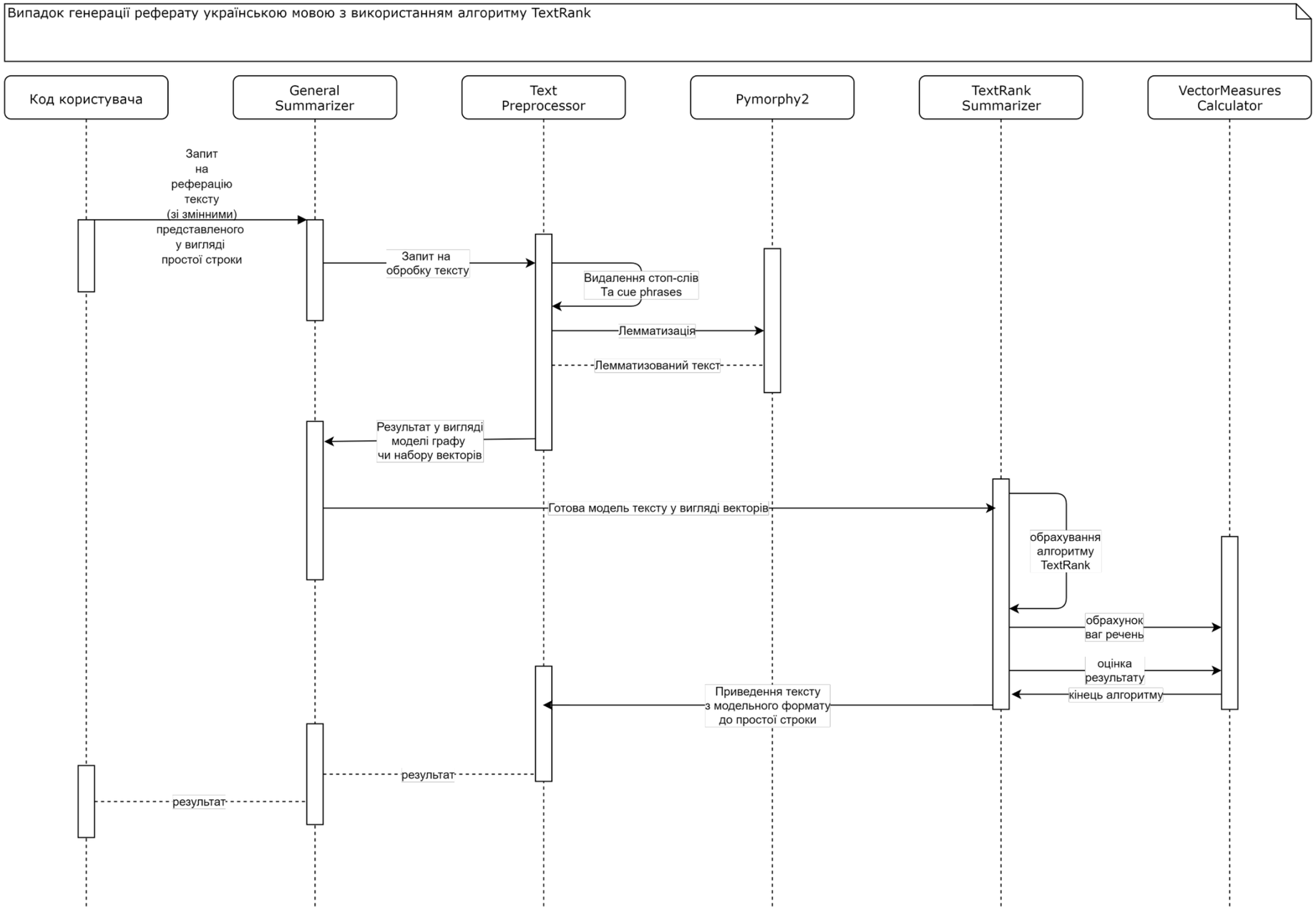
Демонстраційний плакат до магістерської дисертації

Діаграма класів розробленого застосунку

Виконав студент гр. ІП-з81мп Аршакян Г.Д.

Керівник Олійник Ю.О.

ДОДАТОК Г - Діаграма послідовностей



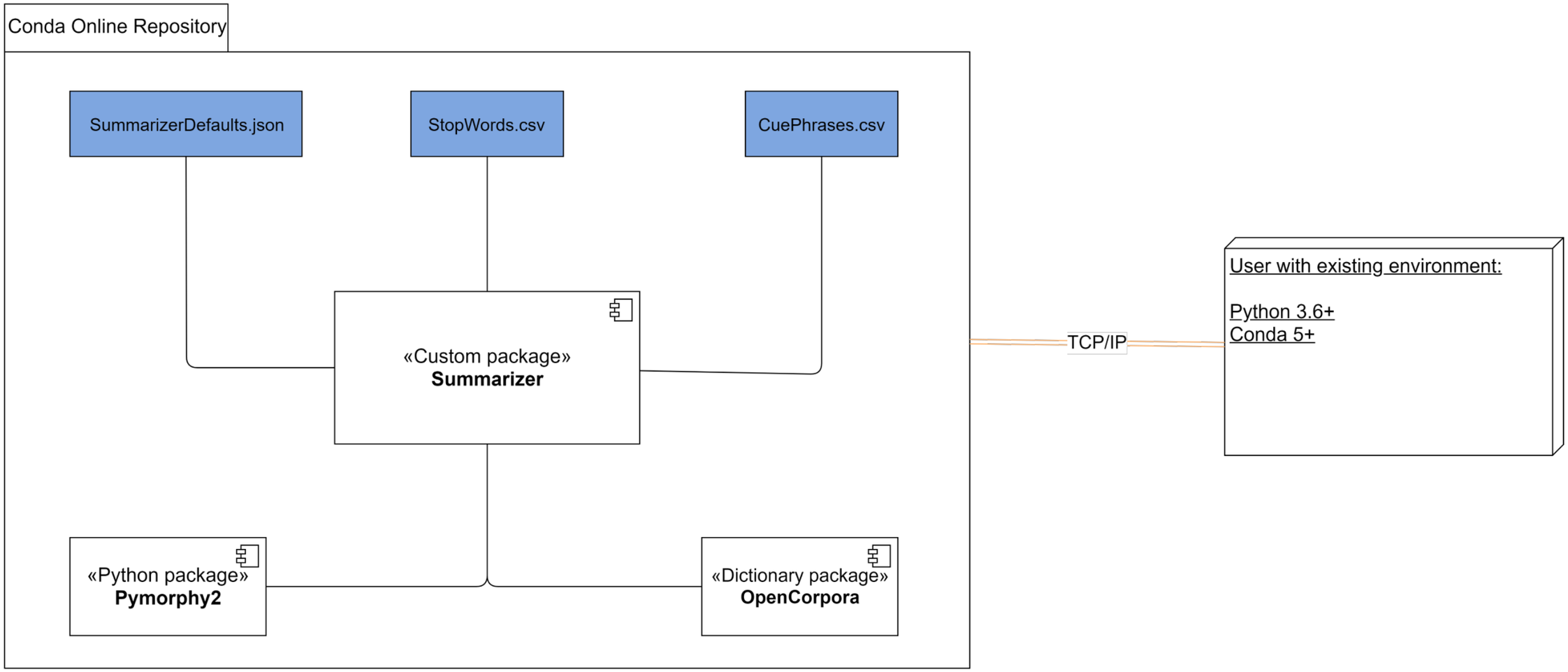
Демонстраційний плакат до магістерської дисертації

Діаграма послідовностей

Виконав студент гр. ІП-з81мп Аршакян Г.Д.

Керівник Олійник Ю.О.

ДОДАТОК Д - Діаграма розгортання розробленого застосунку



Демонстраційний плакат до магістерської дисертації

Діаграма розгортання розробленого застосунку

Виконав студент гр. ІП-з81мп Аршакян Г.Д.

Керівник Олійник Ю.О.